

IBM® Personal Systems

Developer

TECH SUPPORT

A Publication of the IBM Developer Assistance Program
Summer 1991

- Windows vs. OS/2 2.0
- Migrating to OS/2
- Spotlight on Adobe



IBM® Personal Systems Developer

Table of Contents

■ Editor's Comments	3
■ Developer Assistance Program	
Developer Assistance Program Update	4
32-Bit Expedite Program	6
A Look Back at the First DOS	7
■ Spotlight	
Adobe Type Manager® Software in OS/2 1.3.....	9
■ Software Tools	
Cross-Platform Development	19
A Porting Primer	23
Migrating Windows Applications to OS/2 Using WLO	32
Porting Real-World Windows Applications to OS/2 PM	39
Developing PM Applications with Gpf	48
■ 32-Bit OS/2	
OS/2 2.0 Considerations	55
■ Presentation Manager	
System Input Hooks under OS/2	72
■ Application Enablers	
IBM SAA Networking Services/2	82
Multiple Conversation OS/2 Server Using IBM SAA	87
■ Extended Edition	
Client-Server Solutions: OS/2 or Windows 3.0?	92
■ LAN Support	
LAN Application Certification Support	101
■ OS/2 Multi-User	
Remote-OS Multi-User System for OS/2	104
■ International	
OS/2: The International Scene	115
■ Systems Application Architecture	
Multi-Phased Cooperative Processing Application Development	119

The *IBM Personal Systems Developer* is published quarterly by IBM Software Developer Support, Internal Zip 2212, 1000 NW 51 Street, Boca Raton, Florida 33429. Phone (407) 982-6408, Fax (407) 443-4233. IBM employees and M&S branch office customers can subscribe to the *Personal Systems Developer* through IBM Mechanicsburg's Systems Library Subscription Service (SLSS) using the Developer's order number, G362-0001. Others can subscribe by calling the publisher, Graphics Plus Inc., directly at (800) Read -OS2. Subscriptions (U.S. only) are \$39.95 yearly. Questions, suggestions, and article ideas should be sent to the Editor.

While back issues are not generally available, articles from the first seven issues of the *Personal Systems Developer* have been published in *OS/2 Notebook: The Best of the IBM Personal Systems Developer*. This book can be bought at a local bookstore (\$29.95) or by calling Microsoft Press at (800) MS-Press. Its Mechanicsburg order number is G362-0003-00.

Editor: Dick Conklin, IBM Software Developer Support. Publisher: Graphics Plus, Inc., 640 Knowlton Street, Bridgeport, CT. 06608, Project Manager Jo-Ann Radin.

© Copyright 1991 by International Business Machines Corporation. Printed in U.S.A.

IBM Personal Systems Developer is published by the Entry Systems Division of International Business Machines Corporation, Boca Raton, Florida, U.S.A., Dick Conklin, Editor.

Titles and abstracts, but no other portions, of information in this publication may be copied and distributed by computer-based and other information service systems. Permission to republish information from this publication in any other publication or computer-based information system must be obtained from the Editor.

IBM believes the statements contained herein are accurate as of the date of publication of this document. However, IBM hereby disclaims all warranties either expressed or implied, including without limitation any implied warranty of merchantability or fitness for a particular purpose. In no event will IBM be liable to you for any damages, including any lost profits, lost savings or other incidental or consequential damage arising out of the use or inability to use any information provided through this publication even if IBM has been advised of the possibility of such damages, or for any claim by any other party.

Some states do not allow the limitation or exclusion of liability for incidental or consequential damages so the above limitation or exclusion may not apply to you.

This publication may contain technical inaccuracies or typographical errors. Also, illustrations contained here may show prototype equipment. Your system configuration may differ slightly.

This publication may contain articles by non-IBM authors. These articles represent the views of their authors. IBM does not endorse any non-IBM products that may be mentioned. Questions should be directed to the authors.

This information is not intended to be an assertion of future action. IBM expressly reserves the right to change or withdraw current products that may or may not have the same characteristics or codes listed in this publication. Should IBM modify its products in a way that may affect the information contained in this publication, IBM assumes no obligation whatever to inform any user of the modification.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming or services that are not announced in your country. Such references or information must be construed to mean that IBM intends to announce such products, programming or services in your country.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever.

All specifications are subject to change without notice.

To correspond with the *IBM Personal Systems Developer*, please write to the Editor at IBM Corporation, Internal Zip 4607, P.O. Box 1328, Boca Raton, FL 33429-1328.

Trademarks

IBM, PS/2, Operating System/2, OS/2, OS/2 EE, DisplayWrite, Audio Visual Connection, AIX, AS/400 NetView Personal System/2 and AVC are registered trademarks of IBM Corporation.

APPN, Communication Manager, Database Manager, Presentation Manager, OS/2 Application Manager, Systems Application Architecture, SAA, Application System (AS), IBMLink, CICS, DB2, RISC, SQL/DSNetView, LAN Requester, IBM COBOL/2, CUA, Common User Access, MVS, IBM C, and NetBIOS are trademarks of IBM Corporation.

Microsoft, MS, MS-DOS, Excel, CodeView, WORD, LAN Manager, and Windows are registered trademarks of Microsoft Corporation.

Apple, Mac and Macintosh are registered trademarks of Apple Computer, Inc.

DeScribe is a registered trademark of DeScribe, Inc.

UNIX is a registered trademark of UNIX System Laboratories, Inc.

Lotus, Freelance and 1-2-3/G are registered trademarks of Lotus Development Corporation.

Remote-OS, HOSTESS, Software Lifeline Inc. and OCTOPORT are registered trademarks of Software Lifeline Inc.

DEC, VAX, VT220, and DECnet are registered trademarks of Digital Equipment Corporation.

MultiVendor Architecture is a trademark of SAS Institute Inc.

Novell, Novell Netware Requester and Netware SQL are registered trademarks of Novell, Inc.

EASEL is a registered trademark of EASEL Corporation.

Adobe Type Manager and PostScript are registered trademarks of Adobe Systems Incorporated.

Corel is a registered trademark of Corel Systems Corporation.

Arity is a registered trademark of Arity Corporation.

Object/1 and mdbs are registered trademarks of mdbs, Inc.

System Architect is a trademark of Popkin Software & Systems, Inc.

XVT (The Extensible Virtual Toolkit) is a trademark of XVT Software, Inc.

AutoCAD is a trademark of Autodesk, Inc.

3-Com is a registered trademark of 3-Com Corporation.

Motif is a trademark of Open Systems Foundation.

Ashton-Tate, Framework and dBase are registered trademarks of Ashton-Tate Corporation.

Microfocus Cobol/2 Compiler and Microfocus Cobol/2 Workbench are trademarks of Microfocus Inc.

Realia is a registered trademark or Realia, Inc.

TopSpeed is a registered trademark of Jensen & Partners International, Inc.

Watcom C is a trademark of Watcom Systems Inc.

CASE PM and CASE Works are trademarks of Caseworks, Inc.

WinPro/PM is a trademark of Xian Corporation.

Actor and The Whitewater Group are registered trademarks of the White Water Group.

Choreographer is a registered trademark of GUIDance Technologies, Inc.

Smalltalk/V is a registered trademark of Digitalk, Inc.

386 SX is a trademark of Intel Corporation.

FORMATION DESKTOP is a trademark of Aspen Scientific.

Micrografx Mirrors, Windows "Draw!", In*Vision, Micrografx Designer, and Charism are trademarks of Micrografx, Inc.

PC Draw is a registered trademark of Micrografx, Inc.

Metaphor is a trademark of Metaphor Computer Systems Corporation.

Open Software Foundation and OSF are trademarks of Open Software Foundation.

AMS MicroTradeLine is a trademark of American Management Systems, Inc.

Editor's Comments

As OS/2 Version 2.0 is prepared for general availability, interest in the product is on the rise. Some of our early Developer articles about OS/2 2.0 were reprinted in the OS/2 Notebook (Microsoft Press, 1990). In this issue we're taking a fresh look at the product with Claus Makowka's comparison of OS/2 .0 and Windows 3.0 features. In addition, Dan Harkey and Bob Orfali return to compare these products from the viewpoint of the client-server programmer.

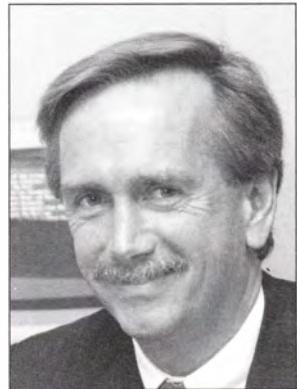
Our OS/2 Application Conversion Workshops continue to receive rave reviews. So, we asked two of the workshop instructors to share their secrets with us. David Moscowitz examines the migration from DOS to OS/2 and Phil Spencer uses Microsoft's Windows Libraries™ for OS/2 (WLO) to get from Windows to PM. Next, Micrografx's Richard Merrick uses Mirrors™ for OS/2 porting and talks about his company's work with IBM on OS/2 Version 2.0.

IBM's new SAA Networking Services/2™ promises to make life a lot easier for OS/2 Extended Edition programmers. We'll look at this product from the vantage of two IBM NS/2 developers and one of our customers who beta-tested it.

This issue's Spotlight article is on Adobe™ — the company who brought their outline fonts to OS/2 Version 1.3. If your applications don't take advantage of this unique feature, here's your chance to reconsider.

There aren't many people who can claim to have worked on every version of DOS and OS/2, but Mel Hallerman is one. We talked to the PC's original chief programmer about the historical events of just ten years ago and his view of the future.

Are you an early user of OS/2 2.0? Have you programmed to the 32-bit API? Do you find that your DOS, Windows and OS/2 1.X programs run better? Do you have a success story to tell or tips and techniques to share? We're compiling the success stories of our OS/2 2.0 beta developers, and we plan to do articles on as many as we can. I'd like to hear from you — my fax number is (407) 433-4233.



Dick Conklin

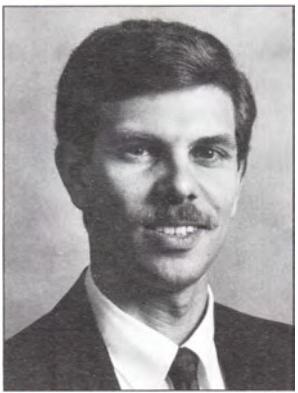


Dick Conklin, Editor



Developer Assistance Program

Developer Assistance Program Update



Joe Carusillo

by *Joe Carusillo*

NEW SERVICES

With IBM's aggressive OS/2 2.0 launch, things are really starting to heat up for us here at the Developer Assistance Program. We have been working on a broad set of new offerings that directly address a number of the requests developers have given us over the last six months. I am very pleased to say that the first of these is an early code program for OS/2 Version 2.0. By the time you read this, all DAP members will have already received an announcement of the program by mail.

*The focus is
on OS/2 2.0*

This new program offers access to, and technical support for a series of beta code drivers. The first of these drivers was made available at the end of June. It was updated in July and will continue to be updated until the driver has the complete functionality of the final product. There is a small charge for each level of the driver. If you are not currently participating, you can enroll at anytime during the program. Our goal for the early code program is to provide software developers an opportunity to do some early compatibility testing of their existing DOS, Microsoft Windows, and OS/2 16-bit applications. We have done extensive compatibility testing of OS/2 2.0 ourselves and are now looking for developer feedback. If you are not a current member of the DAP program, or just missed our mailing, and would like to participate in this program, you can contact us at (407) 982-6408 for further information.

For those who are ready to exploit the functionality of OS/2 2.0 with a new 32-bit application, we have announced another new program called 32-Bit Expedite. This program provides early beta copies of OS/2 2.0 and the tools you will need to develop new 32-Bit applications. Again all DAP members should have received a mailing announcing the availability of this program. If you missed it or need further information about the 32-Bit Expedite Program, see Walt Tanis's article also in this issue.

We are also in the process of updating our Migration Workshops to support OS/2 2.0. These include support for the following migration paths: DOS to OS/2, OS/2 Text to Presentation Manager, and Microsoft Windows to OS/2 PM. We are also adding a new workshop that covers migrating from 16-bit PM to 32-bit PM. These are five day hands-on porting sessions targeted at API level programmers who are just starting to migrate existing applications (DOS, Windows, OS/2 16-bit) to exploit OS/2 2.0. For more information on the availability of these Workshops, call Sandy Garrison at (407) 982-7920.

Another new program, initiated since the last issue, is the OS/2 Information Exchange on CompuServe. This is an IBM sponsored forum open to anyone with a CompuServe User ID. Its purpose is to establish a public forum for the exchange of information, ideas, suggestions, and helpful hints related to OS/2. If you do not have a CompuServe ID, and would like to see what OS/2 Information Exchange is all about, contact CompuServe directly at (800) 848-8990.

The last program I want to highlight in this issue is the OS/2 Technical Seminars. The first, held in Dallas in late May, was a tremendous success. The focus this year—as you might expect—is on OS/2 2.0. There is one more seminar scheduled here in North America before they travel overseas. For more information on the dates and locations see the description below.

ELIGIBILITY AND ENROLLMENT

The IBM Developer Assistance Program is for software developers working on products for commercial release. There is no charge for enrollment. Participation is open to developers who:

- Develop products that support IBM OS/2 or IBM DOS
- Are a U.S. company or the U.S. subsidiary of a non-U.S. company
- Are currently marketing their own products

Developers not currently marketing a product may qualify by submitting a non-confidential, detailed business plan showing schedules, marketing, and development activities. IBM reserves the right to accept or reject an application based on this business plan.

To request an application form or ask questions about the program, contact us at:

IBM Corporation
Developer Assistance Program
Internal Zip 2230
P.O. Box 1328
Boca Raton, FL 33429-1328
(407) 982-6408

1991 IBM OS/2 TECHNICAL SEMINARS

The OS/2 Technical Seminar is an excellent chance to get updated on what's new with OS/2—particularly Version 2.0. These week-long seminars are designed for experienced OS/2 software developers (both vendor and corporate customers). They include main tent and elective sessions with demonstrations and time for hands-on. Topics include OS/2 and Windows Considerations, Developer Tools and Toolkits, OS/2 Extended Edition, Performance and Tuning, Multimedia, the 32-Bit API, Presentation Manager and LAN Server Enhancements, and other topics.

The \$1495 (U.S.) seminar fee includes all class sessions, meals (including breakfast and lunch), the seminar workbook, OS/2 Version 2.0 prototype code, sample programs, tools, and documentation. Seminars are planned for Washington DC (July), Southern California (September), Australia (August) and Europe (September).

If you would like information about these seminars (schedule, enrollment deadlines) or for U.S. enrollments, please call (800) 548-2464, Monday through Friday, 8:00-4:40 Central time. In Canada call (800) 465-1234.

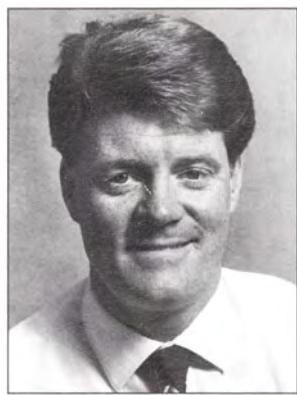
Joe Carusillo, IBM Personal Systems, 1000 NW 51st Street, Boca Raton, FL 33431. Mr. Carusillo is the Manager of Software Developer Programs which includes the Developer Assistance Program. He started with IBM Entry Systems in 1982 and has worked on PC software since that time. He has a BS in Computer Systems Engineering from Columbia University, School of Engineering and Applied Science, in New York City.





Developer Assistance Program

32-Bit Expedite Program



Walt Tanis

This program is for developers of 32-bit OS/2 applications

by Walt Tanis

As part of our continuing commitment to OS/2, IBM has announced the OS/2 32-Bit Expedite Program. This program is designed to fill the needs of software developers who are developing 32-bit OS/2 applications.

As an OS/2 2.0 32-bit software developer you can receive development and marketing assistance from IBM in the following ways:

- Loan of an OS/2 2.0 development environment, including operating system and tools
- Regular updates to the development environment
- Technical support through MCI Mail™ and IBMLINK
- 1-800 service for critical problems
- Online access to OS/2 tools
- Tool product demos
- Priority seating at Migration Workshops
- Priority access to PS/2 loaner systems

In addition, your articles written about 32-bit OS/2 development will be given priority consideration for inclusion in the IBM Personal Systems Developer magazine. Developer demonstrations featuring 32-bit applications will also be the priority focus of all upcoming 1991 business shows.

To be considered for this new program please complete and return the OS/2 32-bit Expedite Program application. If you are a DAP member, you should have already received an application form. If you have not received our mailing or would like additional information please contact us. We are looking forward to a new class of OS/2 applications on our new 32-bit platform.

OS/2 32-Bit Expedite Program
1000 NW 51st ST
International Zip 2230
Boca Raton, FL 33431
407-982-6408

Walt Tanis, IBM Personal Systems, 1000 NW 51st St, Boca Raton, FL 33431. Mr. Tanis joined IBM in June 1980. After working three years in IBM hardware procurement in Tucson Arizona, he joined the software contracting team in Boca Raton, FL. He is currently a Software Application Planner in PS Software Developer Support where he provides plan input and strategies directed towards increasing 32-bit OS/2 applications. He has a BS in accounting from State University of New York and an MBA from Corpus Christi State University in Texas.

An Interview With Mel Hallerman

A Look Back at the First DOS



by Dick Conklin

Just ten years ago, IBM announced the original IBM Personal Computer. One of the key players in the development and launch of that historic product was Mel Hallerman, IBM's Chief Programmer on the PC project. We thought our readers would enjoy a look back at that exciting time.

Developer: Mel, what was your role as chief programmer in 1980-81?

Hallerman: Back then there weren't a lot of us, so we filled multiple roles. My main activities were:

- Define the content and specification of PC DOS
- Technical interface to Microsoft
- Technical interface to the Engineering group — mainly on BIOS questions
- Work with our information developers to define documentation content
- Chief Troubleshooter
- Dotted-line technical assistant to Don Estridge, the project's director.

Developer: How did you manage to develop the PC and its software in just one year?

- First, one year of 80-hour weeks is really two years.
- We had a lot of new college-hires who didn't know it couldn't be done. (And, I didn't tell them).

Seriously, we were a team dedicated to bringing out a personal computer. To us this was a holy mission and a golden opportunity. No one had to be asked to work long hours — we wanted to. We watched out for each other and picked up if someone dropped something.

Most people go through their entire lives without having the opportunity to participate in a defining moment such as this. I think all of us who participated in the project are grateful we had a chance to contribute not just to IBM's success but also to define the next step in the Information Age.

Developer: DOS was unheard of before the IBM PC. Why was it selected over a better-known operating system like CP/M?

Hallerman: We did look at CP/M. In addition to contractual/procedural problems there were problems in reshaping it into the form we wanted. We were already working with Microsoft on BASIC and it was very convenient to also do DOS with them. The working relationship was very good.

Developer: How did IBM influence the content of DOS 1.0?

Hallerman: Essentially, we took the existing version of DOS that Microsoft had, and we wrote a specification reshaping it into the way we wanted it to look. As a guess, I would say IBM specified 75% of the changes to the existing DOS that resulted in the PC DOS 1.0 that we finally shipped.

Our goal was non-cryptic commands, English language error messages (instead of error codes), small memory size (12 Kb) and reasonable performance.



Mel Hallerman

*One year of
80-hour weeks
is really two
years*



Developer: DOS has seen several evolutions since version 1.0. Did you ever guess back then that it would have today's level of function?

Hallerman: I would like to appear clairvoyant and say yes, but the answer is no. We were completely focused on the first release and limited our future thinking to the next two years. It is amazing to think it has only been ten years and to see how far DOS has come.

Developer: If you could turn back the clock, what would you have done differently in 1980-81?

Hallerman:

- IBM exclusive ownership of DOS
- More patents on the IBM PC.
- Don't publish the entire BIOS listing, just the entry points.
- Define up-front which aspects of the PC architecture were guaranteed compatibility points and which were not.
- Push for graphics on the monochrome adapter. That way graphics would have been standard for both mono and color. Since the mono was text only, the majority of applications for the first few years were text-based (the lowest common denominator syndrome).

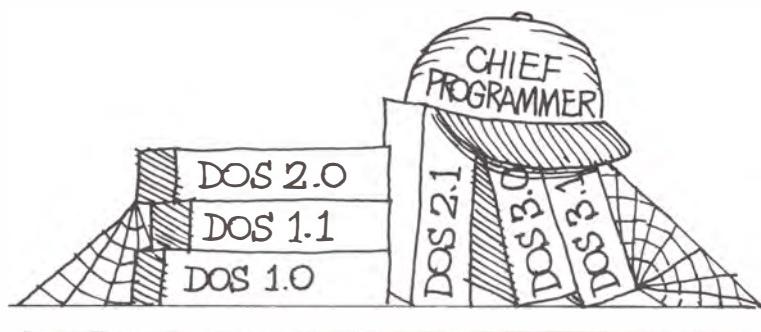
Developer: What is your current role at the Boca Programming Center?

Hallerman: I'm the technical assistant to Tommy Steele, the director of the Boca Programming Center. In addition, I chair the Technology Committee of the Site Technology Council. The Technology Committee is made up of the Senior Technical Staff Members at IBM Boca Raton.

Developer: What is your view of PS/2 operating systems in the future (DOS, Windows®, OS/2, Unix® etc.)?

Hallerman: Hardware is becoming more and more complex internally — busmasters, digital signal processors, advanced graphics processors, memory caches, etc. This functionality will move down through the product line until it is pervasive even in the entry systems. You need a real operating system to mask the complexity and get your money's worth out of the parallelism implied by these hardware advances. I see DOS and DOS/Windows fading away over the next five years. With the ability of OS/2 2.0 and beyond to be a better DOS than DOS, better Windows than Windows plus supporting advanced mission-critical applications, I see OS/2 supplanting DOS/Windows. This is particularly true since the advances in RAM and DASD capacity and cost mean that OS/2 will be able to run on ALL these systems right out of the box.

Unix has a different market and will always play an important role. But, it is not the natural successor to DOS/Windows.



Spotlight

Adobe Type Manager® Software in OS/2 1.3

by Peter Goldstein

Adobe Type Manager® software (ATM) technology has become an integral component of the OS/2 1.3 operating system released last October. This issue's Spotlight provides valuable information about how this technology works within OS/2 and ideas for designing applications that take advantage of its features. During our discussions with the developers at Adobe they presented various challenges, solutions, and visions that most developers can relate to and learn from. ATM dramatically improves OS/2's screen and printer handling capabilities. Combining Adobe Type 1 font outlines and metrics, a rasterizer, and a font cache in virtual memory, ATM provides seamless integration with existing OS/2 applications and device drivers. Compatible with standard OS/2 API calls, it allows developers to integrate advanced font handling capabilities into their applications with minimal effort. For the first time in a PC operating system, there is no difference between screen, system, and printer fonts.

HISTORY OF ATM

Adobe first implemented ATM technology for the Macintosh in 1988. More recently they have ported it to OS/2 and Windows. Although the three products have been built from the same core components, there are some significant differences in how they interface to their respective operating systems.

"A fundamental difference between our OS/2 implementation and the others is that the OS/2 ATM was done with the cooperation of the system vendor as an integral part of the operating system," explained Jon von Zelowitz, a senior member of Adobe's technical staff. "We are this 'uninvited guest' on the Macintosh and in Windows. Because of this, we had to use some very clever programming techniques

to make the operating system do what we expected. With the OS/2 ATM product, we are the operating system and were able to take advantage of that relationship. We were able to have better communications internally with the operating system as a result of it being more tightly coupled."

"The OS/2 ATM also presented a challenge in that we were required to maintain 100% compatibility with existing APIs. We were installing a font system that worked in a completely different way from what was there before, but could not allow the applications to see a difference. That constraint did not exist on other ATM platforms because of the way we had to interface with the operating systems."



*Adobe Type
Manager
(ATM) fonts
are an integral
component
of OS/2*

Adobe Systems Incorporated

Address:	1585 Charleston Road Mountain View, CA 94039 (415) 961-4400
Founded:	1982
Employees:	546
Sales Channels:	OEMs, 3000+ US dealers, 300+ dealers in Europe and Australia, distributors worldwide
Other Products:	PostScript interpreter, Display PostScript, various graphics and type products for DOS, UNIX, Macintosh, Windows, and OS/2 systems
Developer Contact:	PostScript Developers Association (415) 961-4111



"On the Macintosh and Windows platforms, our architecture was designed to intercept font requests to provide ATM service," added Ming Lau, Project Manager. "When you intercept requests, you catch as much as you can. It is really an after-the-fact mode of operation. There will always be some things you can't completely do because of this limitation."

Another design constraint specific to the OS/2 ATM had to do with character sets. The standard Macintosh character set is the same as Adobe's standard Type 1 PostScript® character set. OS/2's standard character set, or code pages, are very different. Adobe had to make sure that the OS/2 system fonts would have all the characters that are standard across OS/2.

ADOBE TYPE 1 FONT STANDARD

Central to the power ATM offers OS/2 applications is the adherence to Adobe's Type 1 font format. Type 1 is simply Adobe's designation for defining font outlines. More than 6500 typefaces are now available from over 30 vendors in the Type 1 format. Adobe alone offers 1000 Type 1 fonts through distributors and resellers. Any Type 1 font purchased in PC format can be installed easily by users through OS/2's control panel and used just as easily as the built-in system fonts.

The outline fonts used by ATM in OS/2 are the same Adobe Type 1 fonts used in other ATM implementations, PostScript Language output devices, and Display PostScript system windowing environments. As a result, display and printer output created on any of these systems can be moved to

another without losing important type formatting information.

As a part of SAA, Adobe's Type 1 font format provides a continuity across multiple IBM platforms. Font names, character widths, and other typographical information specified by an SAA compliant application on one system can be properly recreated from the same outlines and font metrics on another. The Type 1 font format has already been integrated into OS/2 1.3 with ATM, VM with a PostScript Language interpreter, and AIX with Display PostScript. Beyond SAA, Adobe's Type 1 font format is a dominant standard on Macintosh, NeXT, and Windows platforms, providing even more cross-platform continuity.

Type 1 fonts are installed as pairs of files on OS/2 machines. The .PFB file contains character outlines which are individually mapped on 1000 x 1000 grids. Font metrics are stored in a companion .AFM file. This file contains all of the information OS/2 needs to know about using the fonts once they are rasterized. Information including the full font name, character widths, and kerning pairs is stored here.

ATM RASTERIZER

Adobe has three different product families that can rasterize Type 1 fonts. The PostScript Language was the first to market, in 1986, and is a sophisticated interpreter designed primarily for use in print controllers. (See Figure 1.) PostScript consists of three subsystems: Type 1 font rasterizer, graphics handler, and image handler. The Display PostScript system also has these three subsystems, but is designed

	PostScript	Display PostScript	OS/2	
			ATM	GPI
Type scaled or rotated	hard copy	hard copy & display	hard copy & display	
Type as mask	hard copy	hard copy & display		hard copy & display
Graphics (line art)	hard copy	hard copy & display		hard copy & display
Images (halftones)	hard copy	hard copy & display		hard copy & display

Figure 1. Through ATM and GPI, OS/2 Provides All the Services Necessary to Synchronize Screen Output with PostScript Printer Output

for interactive use on the screen. ATM is basically the Type 1 font rasterizer subsystem found in both PostScript and Display PostScript.

"The ATM rasterizer is just the latest generation of PostScript Type 1 Language rasterizers," von Zelowitz said. "There have been many generations. We keep moving ahead with the technology to improve quality and speed. The ATM rasterizer was really a revolutionary move for us. New rasterization algorithms gave us enough speed to put it into an interactive category. In fact, the developments that went into the ATM rasterizer are now being put back into PostScript Language laser printers."

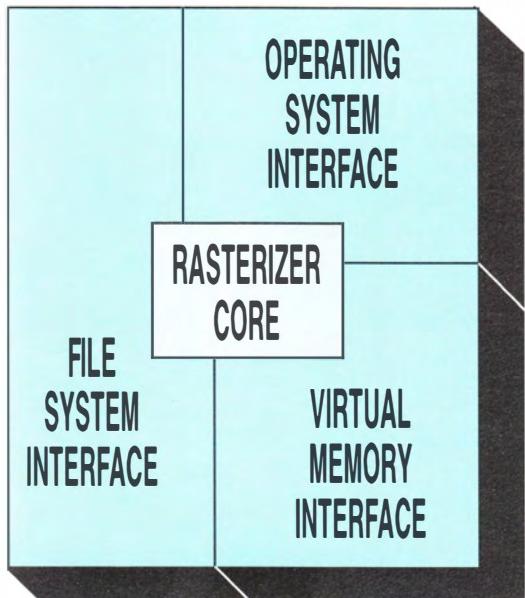


Figure 2. ATM Rasterizer Model

"An important thing to recognize is that we're still working with the same Type 1 fonts we created years ago, even though the rasterization technology has changed over and over. I think that's a good proof of the universal structure that is represented in the Type 1 fonts."

PORTING ATM

About 30 to 40 percent of the ATM model in OS/2 1.3 was developed specifically for OS/2. The remainder of the code is shared with the other ATM implementations. What you find in all ATM implementations is a standard core rasterizer surrounded by a "glue" layer that is customized for the

operating system. (See Figure 2.) This "glue" layer has three segments. One segment interfaces with the operating system, another with memory where the font cache resides, and a third with the file system where font files are stored. The core hardly ever changes, but the "glue" layer changes considerably with the platform. The only exception to this model is the ATM for the Macintosh, where a fourth segment of the "glue" layer has to provide a bitmap cache. On OS/2 the bitmap cache is managed directly by the PM Graphics Engine GRE and does not require ATM to provide it.

The way that the ATM rasterizer is structured, the core can change without having to rewrite the "glue" layer. In OS/2 2.0 the core of the ATM rasterizer will be revised, and will plug into the rest of the system quite easily.

HOW ATM INTERFACES TO OS/2

The heart of OS/2's graphics handling capability is GRE. (See Figure 3.) All OS/2 API calls that deal with character generation are processed by the GRE. When a specific character bitmap is requested by OS/2, the GRE looks in its Bitmap Cache. If the character bitmap is not in the Bitmap Cache, the GRE requests it from ATM. The requests are passed internally to ATM via IBM's Intelligent Font Interface (IFI). This IFI interface is the mechanism through which all OS/2 font handling is performed. It provided an excellent "common ground" for both Adobe and IBM developers to collaborate on, essential for an operating system level product like this. The IFI interface also allows improvements to be made by either

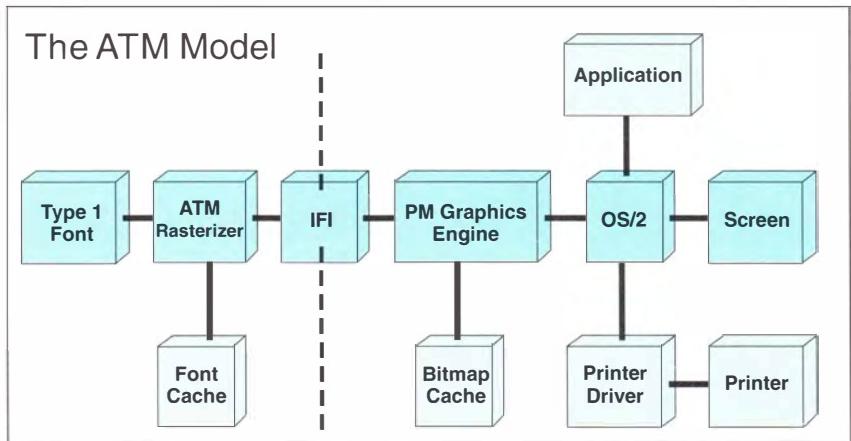


Figure 3. Character Generation Model Employment in OS/2 1.3





There is a lot of activity going on under the hood when you use these fonts

Adobe in ATM or IBM in the GRE without requiring major changes to the other, so long as those changes conform with the IFI specifications.

HOW ATM WORKS ON THE SCREEN

When the PM GRE in OS/2 requires a character bitmap and does not find it in its own cache, it makes a request to ATM. The ATM rasterizer then looks to see if that character's outline and font metrics are available from its font cache. If it is not, ATM opens the Type 1 font files containing the outlines and metrics and reads them in their entirety into font cache. This font cache resides in virtual memory and can grow dynamically as needed. ATM then rasterizes only the requested character with the appropriate attributes and passes it on to the GRE.

As additional characters from the same font are requested, ATM reads them from the

font cache, rasterizes them, and passes them to the GRE. Whenever a previously requested character bitmap is requested again by the operating system, the GRE reads the character from its bitmap cache and passes it to the screen.

"There is a lot of activity going on under the hood when you use these fonts," von Zelowitz remarked. "You can get a feel for just how much activity by watching the screen. When a font is requested for the first time, the display appears to hesitate during the drawing of the first line or two of text. This is due to the extra time required for ATM to read in the font outlines and metrics, rasterize the required characters, and pass the resultant bitmaps to the GRE. Once a character's bitmap is in the GRE cache, however, it can be accessed by OS/2 very quickly."



Jon von Zelowitz

ATM VS. VECTOR FONTS

"OS/2 1.2 offered scaleable fonts in addition to the standard bit-mapped fonts, but they were mostly GPI primitives," explained Lau.

"With OS/2 1.2's scaleable vector fonts, every character was drawn in the same way any other GPI graphic was drawn. When

trying to display a whole page, every letter had to be drawn individually. That was very, very slow. The vector fonts didn't have 'hints' in them either, so anything at 24 point size or below was virtually illegible. Type 1 fonts have 'hints' in them which change the properties of each character when rasterized at small sizes so that they are more legible. When rasterized, scaleable Type 1 fonts behave just as quickly as bitmap fonts did in 1.2, but with much better quality.



Ming Lau

TUNING SYSTEM FONTS

OS/2 1.3 ships with several system fonts provided by IBM: Times New Roman, Helvetica, Courier, and Symbol. Although these fonts are Type 1 compatible, the developers at Adobe did some fine tuning to increase their efficiency.

von Zelowitz explained, "Because these system fonts are going to be used all the time, we created a font file format which is functionally equivalent to the structure in the font cache. System fonts are held in an OS/2 DLL file, allowing us to bring them in very quickly and read them more efficiently. They are always available from virtual memory in this DLL."

"The DLL file format that our system fonts are in is neither published or standard, and is likely to change over time," Lau pointed out. "We do this only to make the limited number of necessary system fonts available more easily out of the box, guaranteeing that the system will always have these fonts available. It would not be wise to write any other Type 1 fonts this way."

HOW ATM WORKS ON THE PRINTER

ATM works much the same on the printer as it does on the screen. Because OS/2 printer drivers interface to the same PM GRE as the screen drivers, the process of requesting, rasterizing, and outputting characters is the same. Text printed using ATM fonts is sent to the printer in graphics mode. It is rasterized by ATM at the resolution specified by the printer's OS/2 device driver. This dramatically enhances the capabilities of all printers when used with their OS/2 device drivers. Without having to rewrite printer drivers, the same high quality Type 1 fonts previously available only on PostScript devices can be output on virtually any OS/2 device with identical metrics (height, width, weight, spacing, formatting).

Even so-called "dumb" printers can be used with ATM and output high-quality Type 1 fonts. Instead of sending screen resolution character bitmaps to the printer, which are typically between 60 and 100 dpi, OS/2 has them rasterized again at printer resolution. In the case of a 9-pin printer this might be 180 dpi. On a laser printer this would usually be 300 dpi. By embedding this capability in the operating system, printer output is guaranteed to align with screen output. And it is accomplished without any special awareness on the part of the application, printer driver, or printer.

ATM AND POSTSCRIPT PRINTERS

When used with Postscript printers, documents created using ATM fonts print faster. With non-Postscript printers ATM has to rasterize fonts for output and a certain amount of system resources are consumed. Screen resolutions are generally between 60 and 100 dpi, so bitmaps do not take very long to create; nor do they require much virtual memory for storage. Printers, especially lasers, offer resolutions up to 300 dpi and greater. Font bitmaps at these resolutions take more time to create and more memory to be stored in than screen bitmaps.

Additionally, shipping high resolution bitmaps via a printer port can take a significant amount of time. When printing to a PostScript printer the GRE does not have ATM rasterize printer resolution fonts. Instead it sends the fonts and instructs the PostScript printer to rasterize the characters

on its own. Passing off this process to the printer both conserves resources on the OS/2 workstation and speeds up transmission of documents to the printer.

ATM AND BUILT-IN PRINTER FONTS

The developers at Adobe made more provisions for printers that do not have PostScript Language capabilities, but do have built-in fonts of their own. In these cases, all ATM fonts are rasterized at the workstation and sent to the printer as printer resolution bitmaps. Fonts that are proprietary to the printer are handled the same way as they had been before ATM. Bitmapped screen fonts, usually supplied by the developer of the printer driver, are first installed via the control panel. Applications requesting these fonts are provided with the character bitmaps by the GRE. Because these fonts are supplied in bitmap format, they are only available in fixed sizes and weights. When it comes time to print, the GRE passes requests for characters and fonts to the printer driver without supplying bitmaps. The printer then uses its own stored bitmaps to create characters on the page. In the case of HP's PCL 5 page description language, which uses outline fonts, the print controller rasterizes the fonts much like PostScript does



Development Team (l-r) Ming Lau, Gary Kuhn, Peter Koester, and Jon von Zelowitz

and produces bitmaps at printer resolution. Screen fonts for this form of outline type are still limited to fixed size bitmaps installed via the OS/2 control panel.



*Competition
was a very
real pressure
to us*

QUESTIONS FOR THE DEVELOPERS

Developer: Was this your first project for IBM?

Lau: No, we've had several projects with IBM over the years. We developed PostScript controllers for the 4216 printer with IBM in 1987, S/370 PostScript for mainframes, and are currently developing Display PostScript for IBM AIX.

Developer: What do you like about programming for OS/2?

von Zelowitz: The fact that it's a virtual memory environment made the design of this code a lot more straightforward. A lot of problems of older style memory systems, like worrying about paging out font caches, are being taken care of very neatly for me. I also didn't have to worry about overlay structures or strange intersegment jumps. There is still, of course, the 64K segment issue in 16-bit OS/2, so the 32-bit 2.0 port is probably going to be more efficient. Overall I was pleased with how OS/2 supported this development.

Developer: How was product testing handled?

Lau: In the early phases we contacted the major ISVs that were writing applications, including Aldus®, Microsoft®, Informix®, DeScribe®, Corel®, Micrografx® and number of others. That was a sort of informal way of testing that helped. Very rigorous system tests were performed at IBM in Boca Raton by Jean Shortley and her staff. Because the operating system and applications already existed prior to testing, the ATM code could be tested from the first day of development until the end. As it turned out, the ATM rasterizer itself was relatively bug free.

Developer: What were your biggest challenges?

von Zelowitz: One was maintaining backwards compatibility to the old way of working with fonts. We had to keep compatible with the APIs OS/2 had always supported. We couldn't break anything.

Lau: From my point of view, competition was a very real pressure to us. ATM had to be fast enough to satisfy users so that they would not look toward third party font management subsystems. The PostScript printer driver was also a very important piece. When we first started the project, the PostScript printer driver written by Microsoft was incompatible with ATM. The best way to resolve the situation would have been to redesign the driver from scratch, but there was not enough time. IBM, with our input, made changes to the driver to satisfy ATM's requirements.

Developer: Can you give us a few examples of how a developer can take advantage of ATM technology?

von Zelowitz: There are many applications which just use the API calls, assuming that the system will provide fonts for them. Those applications immediately take advantage of ATM. Because ATM Type 1 fonts have become system fonts, they start to pop up on your menus and everything works neatly when you use standard OS/2 API calls. DeScribe, PageMaker, and Wingz are excellent examples of this.

Lau: Lotus is very pleased with ATM capability for Freelance. It gives them an extremely high quality look without having to develop custom fonts.

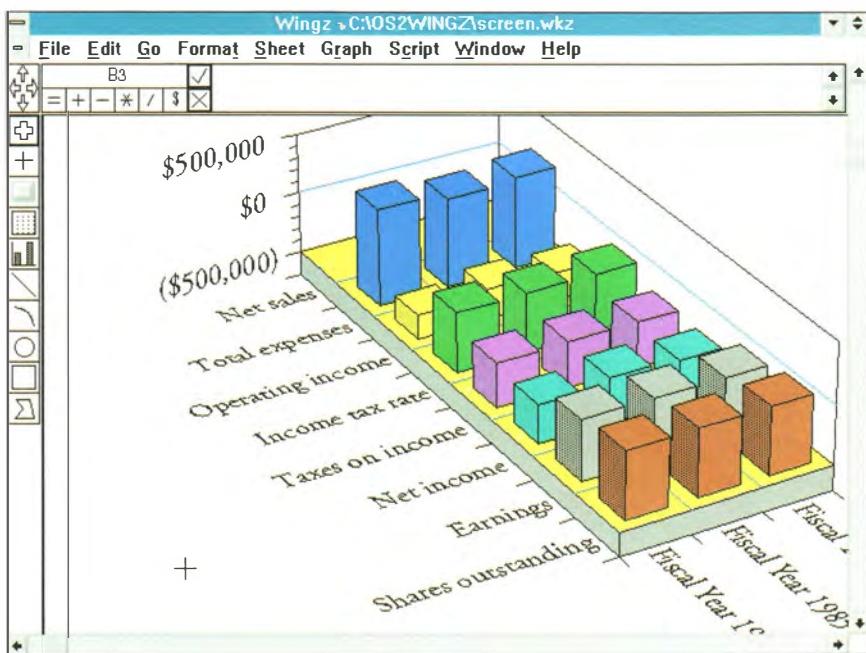


Figure 4. Wingz screen shot with 3D looking text



Gary Kuhn

Kuhn: Wingz has really taken advantage of our font handling capabilities. In the 3D graphics they do you can slant and skew fonts to get more of a 3D feeling. Before ATM you would have to create the fonts as a graphic to do that. With ATM you can just make a

call to the operating system and let it handle the scaling and rotation algorithms. (See Figure 4.)

Lau: Another ATM feature that is very easy for developers to incorporate has already been done by DeScribe. In DeScribe you can arbitrarily zoom however much you want in a document. Describe gives you a slider to choose any incremental zoom instead of the traditional 50 or 75 percent that bitmap fonts can handle.

Kuhn: An applications programmer of, say, a spreadsheet could have a "fit to page" option that would automatically scale a print area to fit on one screen or printed page. This is another good example of how ATM can be used to enhance traditional business applications, not just publishing and presentation applications.

Lau: You can do these things without having to do anything special. This is a very rich graphic environment for a developer. I think that's how developers see this enhancement product. Wingz and DeScribe demonstrate obvious cases of enhanced speed and flexibility by having ATM built into the operating system.

Developer: Does ATM support plotters?

Lau: Yes, just like any other graphics device. Normally, plotters are the hardest devices to create fonts for. With ATM, the fonts are created as bitmaps or vectors, depending on plotter's driver, and are output on the plotter properly.

Developer: Do you have any recommendations for developers new to GUI programming?

Lau: The Vectfont.C program written by Charles Petzold is probably the most significant piece of code that a developer could look at today. I don't know of anything as extensive as that. I use it as a reference. It tests GPI's font capabilities in many ways. That example of coding is exactly how OS/2 calls ATM, using standard API calls. Because ATM is designed so you don't need any extra interface, if you know how to program the GPI, and how to use the GPI font mechanism, you already know how to access ATM.



INTERVIEW WITH JOHN WARNOCK, CHAIRMAN OF THE BOARD AND CEO

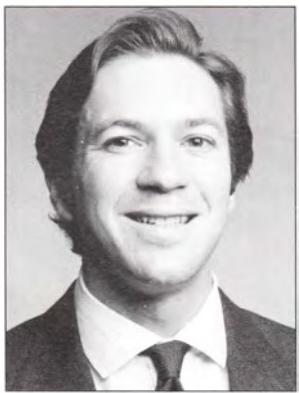
Developer: To what extent are Adobe's products shaped by technology and by marketing?

Warnock: The first level of PostScript was not shaped by any marketing input at all. It was done as a base of experience that was developed by a lot of people working with electronic publishing tools at Xerox's Palo Alto Research Center. The basic language structure was invented before that. It was a combining of those two requirements that defined what PostScript was. It was actually using it over a number of years that refined what the operator set was, so it was mostly by use that defined what the initial product was rather than asking people questions. Adobe has probably the fifth largest graphic arts staff in the Valley, and we use that graphic arts staff to critique our own products, and how they should work and fit together. PostScript Level-2 was defined primarily by sets of user requirements but not by questionnaires or any such marketing studies, mostly in depth discussions with very serious users and what their problems are. Our products have a tendency to be defined by trying to analyze what a customer's problem is and figuring out a solution that fits with the computer as opposed to asking the end user what he wants to see in a computer program.



John Warnock

Our applications really need 32-bit OS/2



Peter Koester

Developer: What is the "Electronic Paper™" concept you have been discussing recently?

Warnock: I think that one of the major problems the industry faces today is that the only common level of interchange that all the computers have with each other is ASCII text. All of the mail systems are based on ASCII text, and this is sort of a jail we've been in for twenty years. We haven't been able to break out of jail. That still has been retained to be the lowest common denominator. I think it's really about time that the industry figured out a way to raise it so that we can send your magazine electronically from one point to another and see it the way that you intend it to be seen with all of the photographs and graphics and images. There is so much information that can not be transmitted in ASCII text now, diagrams, figures, charts, things where we could communicate information much more effectively than we can today, and much more attractively.

Monospaced type is difficult to read. It is very fatiguing. I did a little study one time where a Wall Street Journal would come to you as 500 typewritten pages that you couldn't parse, you couldn't read, you couldn't deal with in any way. You couldn't find your stock, you couldn't do anything with the document. The way newspapers and magazines are formatted really communicate a tremendous amount of information to the reader. People don't always appreciate how much of that is true.

If we can fundamentally solve this document interchange problem, we can significantly expand the market for the usage of computers. A lot of people use computers today to generate documents, but very few people use them to read documents. I think we can expand the base of computer users to a much broader population by having very simple programs that just gain access to information. As networks develop, and inter-enterprise communication gets better and better, this is going to be a preferred way of doing things.

Developer: Have you defined this or offered a standard?

Warnock: No. So far in speeches that we've been giving we're saying this is a major problem in the computer industry and Adobe is going to solve it. We're not going to see if we can solve it, we are going to solve it.

Developer: What is your view of Windows and OS/2?

Warnock: I have great respect for the commitment IBM has to its customers. What they've said is that they made a commitment several years ago to go in a certain direction and customers took that seriously. They started developing programs and making investments based on that stated direction. I think the industry needs to take IBM's efforts around OS/2 very seriously.

Developer: When can we expect other OS/2 applications from Adobe?

Warnock: I think that's an obvious platform that is going to make sense. There are a lot of corporations that have made a commitment, and as those corporations start installing machines that base will be there. Our applications really need 32-bit OS/2. That will make our job a great deal easier. Also, ours happen to be heavyweight applications that require a lot of computing cycles and I think OS/2 is a better long range solution. Windows seems to me to be somewhat opportunistic. I wish it worked better than it does. It does happen to be popular. I think that there will probably be a base of users. But over time, the commitment to customers pays off, and I think it will in IBM's case.

I think our recently announced Multiple Master technology is going to provide developers with a lot of flexibility and tools that they haven't had before. It will provide ways to add very significant value to their platforms. And as OS/2 ATM gains Multiple Master capability, because it is bundled with OS/2, they'll come to rely upon it.

Developer: What kinds of international support do your products offer?

Warnock: PostScript Level 2 Language output devices will all have the ability to handle composite fonts. These include Japanese, Chinese, and other non-Roman languages with very large character sets. Multiple writing directions, and complex encodings are supported by the Type 1 font format.

Our Multiple Master technology will really help in building translated documents, because the user will have much more formatting control. Having parallel French, German, and English translations, for instance, is much easier to control. It saves huge amounts of money. As the application

developers incorporate that capability into their applications, I think we will see the timeliness of foreign language translations and the production of those manuals go much faster. We did a sample where we have English, French, and German. French is in the middle, so if you set for French, set the English a little wide and set the German a little tight, they all take the same amount of space without losing color.

ON MULTIPLE MASTER FONTS

Multiple Masters is a new font technology emerging from the development labs at Adobe Systems. In addition to providing all of the capabilities of Adobe's industry standard Type 1 fonts, Multiple Master fonts offer significantly enhanced capabilities.

Multiple Master fonts begin with the same outline technology as Type 1 fonts. Type 1 fonts contain one outline per character which can be scaled and skewed in any increment. They are limited, however, to the original characteristics of the original outline in terms of weight, width, style and size ratios. Multiple Master fonts are much more dynamic than that. Every Multiple Master font contains four outlines per character; each outline representing a "corner" on a logical grid, as shown in Figure 5. In this example, the four corners contain light condensed, light expanded, black condensed, and black expanded outlines of the letter "B." An infinite number of variations of the character "B" can be created by pointing to a coordinate on the grid. Once an outline with the desired characteristics is selected, it acts just like any Type 1 font. In a like manner, other Multiple Master fonts may contain outlines which define ranges from Light to Bold, Regular to Italic, or Serif to Sans Serif.

The implications of this enabling technology are dramatic. Some of the most significant are described below:

STORAGE

With existing font technologies, each variation of a character outline is stored in a unique font file. A typical font family stores at least regular, italic, bold, and bold italic character outlines. Some font families, like Helvetica™, have so many variations that dozens of font files are required to define the family adequately. With Multiple Master

technology, font families can be consolidated into just a few files while providing an unlimited number of variations. For users who need access to many variations of many font families, the storage savings can amount to tens of megabytes. When dealing with foreign languages like Kanji, which have much larger character sets than Roman languages, storage reductions are of significant value to all users.



SPEED

Printing speed can also be improved when using Multiple Master fonts. Many complex PostScript print jobs require many fonts to be downloaded to the printer. Because Multiple Master fonts act to consolidate font families, fewer font files will need to be transmitted to the printer.

FONT SUBSTITUTION / DOCUMENT PORTABILITY

Operating systems like OS/2 with ATM already have a fairly good means of font substitution. Whenever a document specifies a font not found in the system, it is replaced with a system font of the same size. What happens, though, is a complete reformatting of the document with the system font's metrics. This process can alter

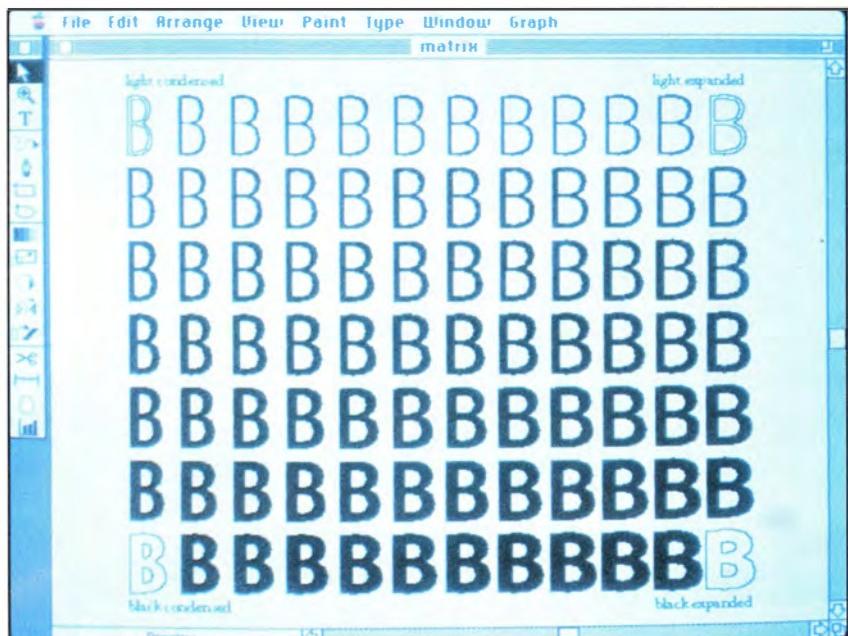


Figure 5. Multiple Master Font Matrix



a document significantly, changing page breaks and line endings, causing clipping of characters and text strings, and decrease the readability of the text. With Multiple Master fonts installed in the system, a font substitution will also take place. But instead of altering the look of the screen or printer text, Multiple Master fonts will emulate the look of the original font. A font rasterizer will select coordinates on the system font's grid that resemble the color of the non-existent font. The "color" of a font is a subjective term used to describe how a block of text appears when read and is a function of weight, scaling, and line styles. This capability will make it easy to interchange documents across systems. No longer will

developers and users have to be concerned with specifying only those fonts which are available on all of the target systems.

COPYFITTING

Another advanced feature of Multiple Master fonts is their capability to be copyfitted more extensively than existing font technologies. Because of the grid system, character outlines can be changed in very subtle increments. Figure 6 illustrates Multiple Master fonts being used to copyfit contextually identical blocks of text in French, English, and German. Notice how the text fits into the same space and has the same color, even though the columns contain varying character counts. This capability will allow for much easier multilingual application development, as screen and print regions will be able to maintain the same boundaries in all languages.

English

The Adobe Type Library
Adobe's PostScript technology is at the heart of each of the more than 1,000 faces in the Adobe Type Library. Because the Berthold Exclusive fonts are rendered in the PostScript Type 1 software format, you can use them with a Macintosh or PC. Then you can print your page on a laser printer, imagesetter or film recorder in a range of resolutions in black-and-white and in color. Within the Adobe Type Library you'll also find the Adobe Originals™, an exciting and rapidly growing part of Adobe's library.

German

Die Adobe SchriftentwörflibRARY.
Jede der mehr als 1.000 Schriften in der Adobe Bibliothek tragen Adobe PostScript Technologie in sich. Auch die Berthold Exclusive Reihe wurde jetzt ins PostScript Type 1 Format übertragen und kann so auf jedem PC, egal ob Macintosh oder IBM, verwendet werden. Die Ausgabe kann dann über Laserdrucker, Fotosatzbelichter oder Filmrechner erfolgen - in Schwarz-Weiß oder Farbe und in den verschiedensten Auflösungen. Ein wunderlicher und rasch wachsender Bereich der Adobe SchriftentwörflibRARY ist die mit viel Liebe zum Detail gestaltete Reihe der Adobe Originals.

French

La typographie d'Adobe
La technologie PostScript d'Adobe constitue le cœur de chacune des 1000 polices de la typographie d'Adobe. Les polices de caractères Berthold Exclusive étant au format PostScript Type 1, vous pouvez les utiliser sur un Macintosh ou sur un PC. Ce qui vous permet ensuite d'imprimer votre document soit à l'aide d'une imprimante laser ou d'une photocomposition (ou "fotocompo"), dans un large choix de résolutions, en noir et blanc ou en couleur. Dans la typographie d'Adobe, vous trouverez également les polices Adobe Originals, une nouvelle gamme intéressante et diversifiée de polices de caractères d'Adobe.

Figure 6. Contextually Equivalent Multilingual Text with Multiple Master Technology

OS/2 ATM Development Team

At Adobe Systems Technical Director:	Tom Malloy
Project Manager:	Ming Lau
Senior Developer:	Jon von Zelowitz
PC Fonts Product Manager:	Peter Koester
At IBM Hursley Technical Director:	Ian McCallion
Project Manager:	Colin Powell
Developers:	Robin Speed Andrew Skates
Integration & Testing:	David Kerr

Software Tools



Cross-Platform Development

by Brian Proffit

Many developers are interested in both OS/2 and Windows, but have limited resources. This column explores answers that can provide both worlds without twice the effort.

I've attended several computer shows in the last few months, and I'm continually amazed at the number and quality of development tools available.

As I talked to the developers early in the year, it became clear that a number of companies out there were uncertain how to proceed in the market. The "Battle of the GUIs" (grapical interfaces) made them hesitant. Because there are always more ideas than there are programmers, many companies just can't consider simultaneous development projects on Windows™ and OS/2™ (and Motif™ and whatever else comes along). Clearly some method has to be found that allows development for one platform to be usable to some degree on other platforms.

The concept of reusable code has been around for a long time. Companies have been developing and marketing standardized subroutine libraries for years. Object-oriented programming carries that concept forward with the use of class libraries whose properties can be inherited by newly developed code. The same concept can also be applied across operating systems, using tools with common front ends, but back ends tailored to the individual platforms. There are several such tools available for OS/2 for which Windows compatible versions also exist.

COMPILERS

The most basic level of compatibility tool is the compiler. Compilers that are source-compatible across platforms can allow many programs to be moved simply by recompiling them. Unfortunately, in the world of Presentation Manager™ and Windows, there are fewer and fewer programs that stick to the compiler interface and its libraries. Most programs make calls to the underlying operating system APIs. Still, there are a number of programs that can gain a new life simply through recompilation. Some of the compilers which have compatible versions for Windows or PM development are:

Arity®/Prolog
Arity Corporation
29 Domino Drive
Concord, MA 01742
Ph. 508-371-1243
FAX 508-371-1487

IBM C/2 1.1™
IBM Corporation
1133 Westchester Ave.
White Plains, NY 10604
Ph. 800-426-2468

Lattice C
Lattice Corporation
12500 S. Highland Ave., Suite 300
Lombard, IL 60148
Ph. 312-916-1600



Brian Proffit

Some tools use reusable code for multiple-platform applications



Microsoft® C 6.0
Microsoft COBOL
Microsoft Corporation
One Microsoft Way
Redmond, WA 98502
Ph. 206-882-8080
FAX 206-883-8101

Microfocus Cobol/2 Compiler™
Microfocus Cobol/2 Workbench™
Micro Focus, Inc.
2465 East Bayshore Rd., Suite 400
Palo Alto, CA 94303
Ph. 415-856-4161
FAX 415-856-6134

Realia COBOL®
Realia, Inc.
10 S. Riverside Plaza
Chicago, IL 60606
Ph. 312-346-0642
FAX 312-346-4638

Stony Brook Professional Modula-2
Stony Brook Software
187 E. Wilbur Road, Suite 9
Thousand Oaks, CA 91360
Ph. 800-624-7487
FAX 805-496-7429

TopSpeed® C
TopSpeed C++
TopSpeed Module-2
TopSpeed Pascal
Jensen & Partners International, Inc.
1101 San Antonio Rd., Suite 301
Mountainview, CA 94043
Ph. 800-543-5202

Watcom C™
Watcom
415 Phillip St.
Waterloo, Ontario, Canada, N2L 3X2
Ph. 800-265-4555
FAX 519-747-4971

Zortech C++ 2.1
Zortech, Inc.
4C Gill Street
Woburn, MA 01808
Ph. 800-848-8408
FAX 617-937-0793

PROTOTYPER/SOURCE GENERATORS

The development of the user interface can be time-consuming in the increasingly complex GUI environment. The use of a prototyping tool can greatly increase programmer productivity. These tools have the additional advantage of allowing the end user to participate in the design process. These tools store the layout design in an intermediate format. That format is common between Windows and PM versions, allowing generation of GUI-oriented pieces of your application, such as the action bar and pulldowns, to be designed just once for both systems.

Source code can then be generated for either of the two environments. Putting these tools together with one of the compilers mentioned above can provide access to both worlds.

Applications Engineer
LBMS
1800 West Loop South, Suite 1800
Houston, TX 77027
Ph. 713-623-0414
FAX 713-623-4955

CASE:PM™ /CASE:W™
Caseworks, Inc.
1 Dunwoody Park, Suite 130
Atlanta, GA 30338
Ph. 404-399-6236
FAX 404-399-9516

Instant Windows
WinSoft
10 Coleman Place, Suite 3
Menlo Park, CA 94025
Ph. 415-324-9552
FAX 415-324-9580

Object Plus
EasySpec Group
17629 El Camino Real, Suite 202
Houston, TX
Ph. 713-480-3233
FAX 713-480-6606

WinPro/PM™
Xian Corporation
625 N. Monroe St.
Ridgewood, NJ 07450
Ph. 201-447-3270
FAX 201-447-2547

HIGH LEVEL INTERFACES

Even greater improvements in portability and often in programmer productivity can be gained through use of a higher level interface. These interfaces shield the developer from the intricacies of the Windows and PM API's by providing a high-level set of callable functions for which they provide backend translations across multiple platforms.

Mewel
 Magma Systems
 15 Bodwell Terrace
 Millburn, NJ 07041
 Ph. 201-912-0192
 FAX 201-912-0103

XVT (The Extensible Virtual Toolkit)™
 XVT Software, Inc.
 1800 30th St., Suite 204
 Boulder, CO 80301
 Ph. 303-443-4223
 FAX 303-443-0969

Nexpert Object
 Neuron Data
 156 University Ave.
 Palo Alto, CA
 Ph. 800-876-4900
 FAX 415-321-3728

Smalltalk/V™
 Digitalk, Inc.
 9841 Airport Blvd.
 Los Angeles, CA 90045
 Ph. 213-645-1082
 FAX 213-645-1306

Object/1®
 mdb's®, Inc.
 Two Executive Dr.
 Lafayette, IN 47902-0248
 Ph. 317-463-2581
 FAX 317-448-6428

System Architect™
 Popkin Software & Systems, Inc.
 11 Park Place
 New York, NY 10007
 Ph. 212-571-3434
 FAX 212-571-3436



OBJECT-ORIENTED ENVIRONMENTS

Many people believe that the greatest level of developer productivity comes through use of tools that follow the object-oriented paradigm. There are a number of such tools that provide compatible versions across Windows and OS/2.

Actor®
 The Whitewater Group®
 1800 Ridge Ave.
 Evanston, IL 60201
 Ph. 708-328-3800
 FAX 708-328-9386

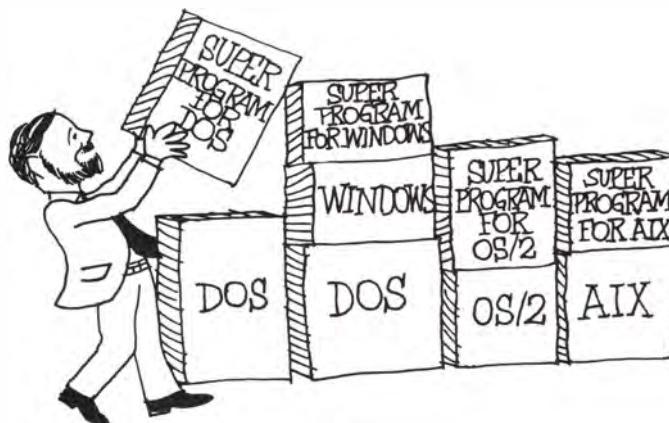
Choreographer®
 GUIdance Technologies, Inc.
 800 Vinial St.
 Pittsburgh, PA 15212
 Ph. 412-231-1300
 FAX 412-231-2076

Easel®
 Easel Corporation
 25 Corporate Dr.
 Burlington, MA 01803
 Ph. 617-221-3000
 FAX 617-221-3099

END USER TOOLS

It could be (and has been) argued extensively whether the new genre of interactive tools is really appropriate for end user development or not. Regardless, a new class of tools is coming into the market, and these tools promise a more interactive, intuitive interface to the user and often to the developer as well.

Toolbook
 Asymetrix
 110 110th Ave. NE, Suite 717
 Bellevue, WA 98004
 Ph. 206-462-0501
 FAX 206-455-3071





SUMMARY

There are trade-offs to consider when selecting an application platform. If you develop for Windows first and port to OS/2, you run the risk of not exploiting OS/2 fully enough to compete with the new generation of OS/2 applications coming into the market. If you develop for OS/2 first, its enhanced function over DOS (even with Windows) can make it difficult to keep the same level of function in a version for DOS extenders. To some extent, selection of a very high level tool shields you from this, but often at the cost of large runtime environments and/or performance penalties.

Clearly the future of personal computing is with systems at the 386SX™ level or higher. The projected sales of systems below that is quite low. There is still a number of 286 systems installed, but that number will continue to dwindle and older hardware has never been the platform for new, innovative applications.

OS/2 2.0 has been available in beta versions for many months now, and will be shipping very soon. It is a superior platform for application development, regardless of the application's target environment. I believe applications that exploit OS/2 2.x's advanced capabilities will command the market share for a long time to come.

Happy Developing!

Brian Proffit, IBM Entry Systems Division, 1000 NW 51st St., Boca Raton, FL 33429. Mr. Proffit is a Senior Programmer in OS/2 software developer strategy. He is currently responsible for OS/2 developer tools. He joined the development laboratory in Boca Raton in 1983 after working as a systems engineer in Dallas.

Software Tools

A Porting Primer

by David Moskowitz

*Portions of this article were extracted from *Converting Applications to OS/2* by Moskowitz, Ivens and Bergman (published by Brady books, copyright 1989 by Productivity Solutions) and a forthcoming revision to that book. Other portions were taken from *OS/2 Power Tools* (by Moskowitz and Ivens, to be published by Brady books) as well as the workshop developed by Productivity Solutions, Inc. All extracted text is used with permission of the copyright holder (Productivity Solutions, Inc. and David Moskowitz).*

Many experienced programmers are justifiably proud of applications they're written for PC-DOS™. But, they've begun to consider other options. Today, they have a choice of personal computing environments including Microsoft Windows and OS/2™.

IBM offers a series of workshops designed to ease the migration from DOS to OS/2, existing OS/2 applications to the Presentation Manager™, and Windows™ applications to PM (for more information see Developer Assistance Program Update, on page 4).

The author of this article is an instructor at these workshops. Below, he summarizes the steps necessary to port DOS-based applications into OS/2. This practical guide will also assist in understanding OS/2 and its relationship to your DOS applications. Because OS/2 offer many advantages not available in DOS, this should be considered a phase in a learning process before moving on to the advanced potential available in OS/2.

WHAT YOU NEED TO GET STARTED

First of all, you'll need OS/2 installed on your Intel 80286-, 80386-, or 80486-based microcomputer. In addition, you'll also need:

- A compiler that supports OS/2 (it should run in OS/2 and provide protected mode libraries).
- The IBM OS/2 Programmers Toolkit for OS/2 1.3 (or equivalent from another vendor).
- Tools for OS/2 including a protected mode editor, linker, make utility, source code control system and anything else you need to make the environment comfortable.

WHAT YOU'LL NEED TO CONSIDER

OS/2 is a protected mode operating system that does not allow protected mode programs to gain exclusive use of the computer. This becomes important when the user wants to do true multitasking. For the programmer, this means an important part of the conversion process is a change in attitude or mindset.

This new mindset was summed by one attendee at the Developer Assistance Workshop: "In DOS my primary concern was to design my programs to keep the system busy. In OS/2 that doesn't work, instead I want to design my programs to keep the user busy."



David Moskowitz

An important part of the conversion process is a change in attitude or mindset



OS/2 VOCABULARY

Any operating system has its own vocabulary. Understanding OS/2's underlying concepts is important because your code will have to address them. In addition, some of the underlying programming paradigms from DOS are no longer applicable in the new environment of OS/2.

The language that we use to describe the system also colors our thoughts about the design and implementation process of writing programs for the system. As we begin to understand the concepts, we can begin to apply them as we write applications that will interact correctly with the power of OS/2.

Process– Under DOS a process is a program that accesses memory and files directly. On the other hand, OS/2 distinguishes between resource ownership and execution. The term process defines the resource owner while execution is accomplished by threads. The execution environment is set up when you establish a process and is shared by all the threads associated with the process.

Threads– All programs have one thing in common—they're made up of a sequence of instructions that are loaded into RAM, read by the CPU and executed. As the CPU moves through each instruction, it is “executing a thread.” OS/2, unlike DOS, permits multiple threads of execution within a single process. An OS/2 program can execute in more than one part of your code at the same time. Each process has at least one thread that may start other threads or processes.

Time slices– Integral to OS/2 is its ability to control and direct. Remember, regardless of the multitasking power of OS/2, the computer has only one CPU which can really only perform one function at a time. Multi-tasking is a kind of “sleight of hand” in which it appears the computer is doing more than one thing at a time. In actuality, the operating system executes one thread for a specific period of time, then it pre-empts

the thread and switches to the next thread eligible to run. The execution interval is called a time slice. OS/2 gives each thread a maximum interval or time slice then it switches to the next thread.

Each time OS/2 switches from one thread to another, it remembers the state of the one it just left behind. When it is time for the first thread to have its turn again, OS/2 restores everything so that it appears as if there had been no interruption. This ability to save a thread's state in memory before moving on to pick up the next in line is called context switching.

Scheduling– The mechanism that OS/2 uses to decide which thread executes next is called scheduling. OS/2 has four classes of priority, three of them (time critical, normal, and idle) each have 32 levels. The fourth (called fixed high) is reserved for the current foreground task including all of the threads associated with the application in either the current active PM window or the current screen group.

OS/2 schedules threads that are eligible to run from the time critical class before threads in the fixed high class. Threads in the fixed high class are scheduled before threads in the normal class, which, in turn are scheduled before threads in the idle class. Scheduling proceeds in a round robin fashion within each class. A thread in a higher class that becomes ready to run will be given a time slice before a thread in a lower priority class.

The scheduler preempts (or interrupts) threads after they have completed their time slice. Each thread is unaware if it has been interrupted or for how long. This means that timing loops that are used in DOS programs will not be accurate or even usable in the OS/2 program.

Memory– OS/2 manages memory through a technique called virtual memory. In general, a program must be loaded into computer's physical memory to execute. In the DOS

world, as many programs as will fit into 640Kb of available memory can be loaded, and the application directly accesses any and all of the memory available. Although real mode is called segmented architecture, it is in reality little more than a series of base register and offset computations, with no segment enforcement. A DOS program can access up to 64Kb from any 16-byte (paragraph) boundary.

In OS/2, as in other operating systems, the program must still be loaded into physical memory before being executed. However, the operating system allows over-commitment of memory, so the sum total of all currently executing applications may exceed the amount of physical memory installed. To support this mode of operating, OS/2 maintains a history of segment usage. If OS/2 needs more memory than is currently available, it looks for the least recently used segments. It swaps data segments to disk and discards code segments. If it needs to restore any of this memory it reloads the data segments from the disk and executable segments from the original executable file. (In OS/2 2.0 the unit of memory management is a 4Kb page.)

There is no memory protection in real mode. In protected mode, if you allocate a 1000-byte segment and try to access byte 1001 the hardware will detect a protection violation. Every OS/2 application is prevented from accessing memory it does not own, and is not allowed to access memory owned by other processes.

Screen groups- In a multitasking environment, we need some way to prevent the output of one program from being confused with another. In OS/2 each full-screen character-based application is given a logical keyboard, screen and mouse. This logical grouping of input/output devices is called a screen group.

THE CONVERSION PROCESS:

The first steps in porting are recompiling and relinking using the protected mode libraries supplied with each compiler. If the original program doesn't contain any DOS specific code and relies only on portable C run time, the task of porting is already complete.

If the programmer retains the .OBJ files for the application, they can skip recompilation and proceed directly to relinking using the protected mode libraries. Applications with DOS specific code, must incorporate the following #include statement to annex the OS/2 header files:

```
#define INCL_BASE
#include <os2.h>
```

INCL_DOS
INCL_SUB
INCL_DOSERRORS

OS/2 kernel definitions (DosBeep())
Video, Keyboard and Mouse definitions
OS/2 file that equates error numbers with symbols.

The #define is required to tell the C preprocessor to draw in all the OS/2 base definitions (function prototypes, structures, macros, etc.) used in kernel (or character) OS/2 programming. In our workshops we recommend that new OS/2 programmers start from this point. As they get more experience they may decide to use the correct subset of include files by changing the #define. Doing this cuts down the compiling time by decreasing the amount of header file information included. Most files will include one or more of the following #define files:

Programmers will also have to choose the right compiler options to produce code suitable for protected mode. If the compiler supports multiple error or warning levels, the one that provides the most information is best. Some errors at debug time can occasionally be traced to compile time warnings that were either not displayed or ignored.





*If your
program
doesn't work
in DOS,
converting it
to OS/2 won't
make it work!*

A more complicated port requires more than a mere recompile. We've found the following five basic steps to be very helpful:

- Fix the segments
- Convert the DOS code to a standard form
- Isolate the system dependent code
- Port the code so that it runs in protected mode
- Convert the code so that it takes advantage of OS/2

The assembly language programmer has to pay attention to all five steps. The high level language programmer may only need to be concerned with the last three.

To port an application we start by fixing the segment references. In DOS we can access up to 64Kb from any paragraph boundary. OS/2 does not allow this assumption. In fact, it requires ruthless adherence to segment boundaries. Since the application can only access allocated memory, code that references hardware specific areas (low memory, interrupt vectors, etc.) will have to be changed, too. Any code that exhibits either of these conditions will have to be rewritten. It will not work in OS/2.

At this point, programmers should also consider changing the code used to circumvent either the ROM BIOS or DOS. In other words, we recommend rewriting code written to get around DOS standards to actually *use* the standard. A conversion table suggesting substitutes for video ROM BIOS functions is available in Figure 1. However, programs which avoided the ROM BIOS used such a wide range of techniques, that no similar table to assist conversion is feasible.

The next step is to rearrange the program so system specific code is isolated in one module. This will make the conversion process easier and simplify testing. In fact,

what we suggest is creating an application level programming interface. In this case, instead of writing to the system Application Programming Interface (API), programmers design their own layers to call system services. The application then calls their system services layer which, in turn, calls system service. This process will ease the transition to new versions of OS/2.

After these basic changes, it's a good idea to test the DOS code to make sure it still works. If it doesn't work in DOS, converting it to OS/2 won't make it work.

A SMALL EXAMPLE:

In the listings shown in Figures 2 and 3, we present both the DOS and OS/2 versions of a program to display the contents of a directory. These show the particulars that must be changed in order to have the program work correctly in OS/2.

In lines 25 and 26 of Figure 2 we define a couple of words to hold the current disk transfer address (DTA). In DOS, the normal procedure is to get the current DTA, save it, establish our own area, use it, then restore the original DTA information. To port this program correctly we have to look for an equivalent OS/2 function to replace the intdosx() calls at line 36 and 45 that get/set the DTA. In OS/2 a DOSFindFirst() serves in place of both steps. Since OS/2 does not have anything analogous to the DOS DTA, we don't need this code, so, we can just throw it away along with the variables defined at lines 25 and 26.

The next intdosx() call is at line 51. Line 47 is the setup for this call to DOS for the Find First function. When we check the OS/2 reference manuals we discover a direct replacement (Figure 4 shows a table of DOS int 21h functions and their OS/2 equivalent, and Figure 1 is a list for the ROM BIOS int 10h functions) that uses a very different calling convention.



ROM BIOS CONVERSION TABLE

	BIOS NAME	OS/2 API
00	Set Video Mode	VioSetMode
01	Set Cursor Type	VioSetCurType
02	Set Cursor Position	VioSetCurPos
03	Read Cursor Position	VioGetCurPos
Get Cursor Type		VioGetCurType
04	Read Light Pen Position	N/A
05	Select Active DisplayPage	N/A
06	Scroll Active Page Up	VioScrollUp
07	Scroll Active Page Down	VioScrollDn
08	Read Character/Attribute at Cursor	VioReadCellStr
09	Write Character/Attribute at Cursor	VioWrtNCell
0A	Write Character only at Cursor	VioWrtNChar
0C	Write Dot	N/A !!
0D	Read Dot	N/A !!
0E	Write TTY to Active Page	VioWrtTTY
0F	Get Video Mode	VioGetMode
10	Set Palette Register	VioSetState ##
13	Write character string	VioWrtCharStr
Write cell string		VioWrtCellStr
FE	VideoBuffer	VioGetBuf
FF	Update Video	VioShowBuf

Note: Those functions that are marked with ## only apply to EGA and VGA display adapters. The functions marked !! pertain only to graphics.

Figure 1. ROM BIOS INT10h Conversion Table

DOS Register CODE

```

1. #include <dos.h>
2. #include <stdio.h>
3.
4. #define DOS_GET_DTA    0x2f
5. #define DOS_SET_DTA    0x1a
6. #define DOS_FIND_FIRST 0x4e
7. #define DOS_FIND_NEXT   0x4f
8.
9. #define fCARRY      0x0000      /* These 5 lines define */
10. #define fPARITY     0x0004      /* a set of masks that */
11. #define fAUX_CARRY  0x0010      /* can be used to test */
12. #define fZERO       0x0040      /* the flags register. */
13. #define fSIGN       0x0080      /* -- */
14.
15. void main (int argc, char * argv[])
16. {
17. union REGS inr, outr; // Define the input/output regs for int86
18. struct SREGS segr;    // Define the segment regs for int86x
19.
20.
21. struct find_t flist;
22. struct find_t far *pfind; // Create a far pointer, avoid a cast
23.

```

Figure 2. DOS Register Code Example (Continued)

```
24.  
25. unsigned int dta_seg;           // Holds the old DTA segment  
26. unsigned int dta_off;          // Holds the old DTA offset  
27.  
28. char far * fname;            // File name pattern  
29.  
30. if(argc < 2)  
31.     fname = ".*";  
32. else  
33.     fname = argv[1];  
34.  
35. inr.h.ah = DOS_GET_DTA;       // Save current DTA address  
36. intdosx(&inr,&outr,&segr);  
37. dta_seg = segr.es;  
38. dta_off = outr.x.bx;  
39.  
40. pfind = &flist;              // Makes getting the seg/off easier  
41.  
42. inr.h.ah = DOS_SET_DTA;       // Set the new DTA  
43. inr.x.dx = FP_OFF(pfind);  
44. segr.ds = FP_SEG(pfind);  
45. intdosx(&inr,&outr,&segr);  
46.  
47. inr.h.ah = DOS_FIND_FIRST;    // start w/the first file  
48. inr.x.cx = _A_NORMAL;  
49. inr.x.dx = FP_OFF(fname);  
50. segr.ds = FP_SEG(fname);  
51. intdosx(&inr,&outr,&segr);  
52. if(!(outr.x.cflag & fCARRY)) // oops if the CARRY flag ON  
53. {  
54.     do                      // keep looking for more files  
55.     {  
56.         printf("%11s\t\t%ld\n", flist.name, flist.size);  
57.         inr.h.ah = DOS_FIND_NEXT;  
58.         intdos(&inr,&outr);  
59.     }  
60.     while(!(outr.x.cflag & fCARRY));  
61. }  
62. else  
63.     printf("\n      No matching files for %s", fname);  
64.  
65. inr.h.ah = 0xla;              // restore old DTA  
66. inr.x.dx = dta_off;  
67. segr.ds = dta_seg;  
68. intdosx(&inr,&outr,&segr);  
69.  
70. }
```

Figure 2. DOS Register Code Example



RESULTING OS/2 CODE

```

1. #define INCL_BASE
2. #include <os2.h>
3.
4. #define A_NORMAL 0x00 // Normal file - No read/write restrictions
5.
6. void main (int argc, char *argv[])
7. {
8.     FILEFINDBUF flist;    // struct used by DosFindFirst/DosFindNext
9.     char * fname;         // Pointer to a filename
10.    USHORT count = 1;     // Number of files to find
11.    HDIR phdir = 0xffff; // Force OS/2 to assign a new handle
12.
13.    if (argc < 2)
14.        fname = ".*";    // Use *.* if the no pattern
15.    else                  // on the command line.
16.        fname = argv[1];
17.
18.    if (DosFindFirst(fname, &phdir, A_NORMAL, &flist, sizeof(flist)
19.                   , &count, 0L) == 0)
20.    {
21.        do
22.        {
23.            printf("%12s\t %ld", flist.achName, flist.cbFileAlloc);
24.        }
25.        while(!DosFindNext(phdir, &flist, sizeof(flist), &count));
26.    }
27.    else
28.        printf("\nNo matching files for %s.\n", fname);
29.    DosFindClose(phdir);
30. }
```

Figure 3. Resulting OS/2 Code

In DOS, we have to tell the operating system the action we want to perform. The values that specify the function and subfunction must be loaded into the Ax register. In OS/2, we call system services directly using a function call. In other words, in DOS programs the interface to system services is predicated on an assembly language calling sequence of registers and interrupts. In

OS/2, all system services are accessed directly by function call with arguments placed on the stack.

There is one other point we should make. On inspection of the first two listings, OS/2 code is shorter and easier to read than the register equivalent version of DOS programs.

INT21H CONVERSION TABLE

	Service	OS/2 API
0	Program Terminate	DosExit
1	Keyboard Input	KbdCharIn
2	Display Output	VioWrtNChar
E	Select Disk	DosSelectDisk
19	Current Disk	DosQCurDisk
1B	Allocation Table Info	DosQFSInfo
1C	Allocation Tbl Info/device	DosQFSInfo
2A	Get Date	DosGetDateTime
2B	Set Date	DosSetDateTime
2C	Get Time	DosGetDateTime
2D	Set Time	DosSetDateTime
2E	Set/reset Verify Switch	DosSetVerify
2F	Get Disk Transfer Address	N/A
30	Get DOS Version	DosGetVersion
33	Ctrl-Break Check	DosSetSigHandler
36	Get Disk Free Space	DosQFSInfo
39	Create Directory	DosMkdir
3A	Remove Directory	DosRmdir
3B	Change Directory	DosChdir
3C	Create a file	DosOpen
3D	Open a file	DosOpen
3E	Close a file	DosClose
3F	Read a file (or device)	DosRead
40	Write to a file (or device)	DosWrite
41	Unlink a file (Delete)	DosDelete
42	Move file read/write pointer	DosChgFilePtr
43	Change file mode	DosSetFileInfo
45	Duplicate a file handle	DosDupHandle
47	Get the current directory	DosQCurDir
48	Allocate Memory	DosAllocSeg
49	Free allocated memory	DosFreeSeg
4A	Modify allocation	DosReallocSeg
4B	Load a program	DosExecPgm
4C	Terminate a process	DosExit
4D	Get Subprocess return code	DosCWait
4E	Find first matching file	DosFindFirst
4F	Find next matching file	DosFindNext
54	Get verify setting	DosQVerify
56	Rename a file	DosMove
57	Get/Set a file's date & time	DosQFileInfo DosSetFileInfo

Figure 4. A Table of DOS int 21h Functions



CLASSIC ARCHITECTURE

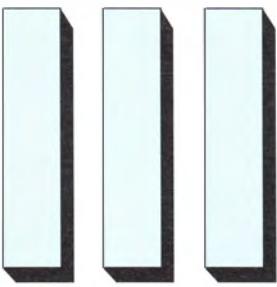


Figure 5. A Representation of DOS Architecture

OS/2 LAYERED ARCHITECTURE

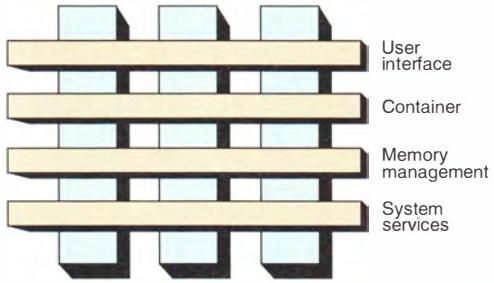


Figure 6. OS/2 Layered Architecture

BEYOND PORTING

After porting code to OS/2 the programmer should consider the last of our five steps: conversion of the code to take advantage of OS/2. This means examining the code with an eye toward adding OS/2 specific features like threads, Interprocess Communication (IPC), Dynamic Link Libraries (DLL's) and Presentation Manager.

Figure 5 shows the classic way of building DOS applications. Each vertical box represents a separate .EXE file. When we needed to build a new program, we used existing LIB's and relinked a new executable file. Good OS/2 programming style dictates that the programmer build horizontal slices across the vertical functionality (see Figure 6). Each slice represents a DLL that encapsulates a particular set of services. In fact, you can apply principles of Object Oriented Programming (OOP) to the design and creation of DLL's—code that is dynamically linked to a program at run time. DLL's act as a server or engine that encapsulates a specific set of services. A consequence of this model is applications built as a series of components that are both server to some parts of the code and clients of other parts. Put all of this together (access to almost unlimited memory, shared reusable code, an operating system that aids the developer) and you have an ideal development platform.

As we tell developers at the DOS to OS/2 workshops, "we don't do any development in DOS for DOS anymore. It takes too long and costs our clients too much money. We even develop Windows applications using OS/2 as a development platform."

David Moskowitz, President of Productivity Solutions, Inc., 164 Avondale Rd., Norristown PA 19403. He was the developer of and remains an instructor for the IBM's OS/2 Conversion Workshops in West Palm Beach, Florida. He has also written a book on the subject of DOS to OS/2 migration, *Converting Applications to OS/2* and authored *OS/2 Power Tools* scheduled for publication early in 1992.

*We don't
have any
development
in DOS for
DOS anymore*





Software Tools

Migrating Windows Applications to OS/2 Using WLO

by Phil Spencer

Phil Spencer, like David Moskowitz (see previous article) has been involved in teaching the Developer Assistance Program's Migration Workshops. The Windows-to-PM workshop uses Microsoft's® Windows™ Libraries for OS/2™ as one way to approach migration issues.

This article will discuss WLO, its operation, use, purpose, strengths and weaknesses.

Microsoft's Widows Libraries for OS/2 (WLO) started life with a different name: the Software Migration Kit (SMK). It went through an alpha and beta release under the SMK title, and was finally released as WLO 0.9 in January 1991.

Around the middle of last year IBM approached QA Training for proposals for a Workshop to cover migration of MS Windows programs to OS/2 PM. The workshops were to attempt to take small numbers of MS Windows programmers and their applications and to convert the applications to OS/2 PM over a 5 day period! QA is well known in Europe for its training expertise in the PC arena including

Phil Spencer



MS Windows and PM — we considered the task for awhile and came to the conclusion that the whole idea would only be viable (in the 5 day period at least) if WLO (then SMK) lived up to our wildest expectations. In the fall of last year the workshop was delivered for the first time based on an alpha release of WLO. WLO performed well, even at that early stage, and the workshop was successful.

WHAT IS WLO?

WLO is a package containing libraries and tools that allow a Windows program to be modified to run under OS/2 PM. Using WLO 0.9 it is generally only necessary to re-link the MS Windows application with the WLO libraries and the program will run under OS/2. There are a number of restrictions and caveats but these are mostly a direct consequence of fundamental differences of operation between MS Windows and OS/2 PM. WLO 1.0 (expected sometime this year) should remove the need for a re-link. This will offer what Microsoft calls a Single Binary Executable that will run without modification on both MS Windows and OS/2 PM. Note that the Single Binary Executable is a new concept which will only be available in applications built with WLO 1.0. It is not the same as the true Binary Compatibility sought by IBM for OS/2 2.xx, which would allow any Windows application to run on OS/2 without modification.

	Platform		
App. Type	MS Win 3.0	OS/2 1.xx	OS/2 2.xx
MS Win 3.0	Yes	WLO	Yes/WLO
OS/2 1.xx	No	Yes	Yes
OS/2 2.xx	No	No	Yes

Figure 1. WLO's Place Between OS/2 and Windows



The table in Figure 1 shows how WLO fits into the OS/2 and Windows jigsaw. It shows application types that run on the various platforms.

System	Min Processor
MS Win 3.0	8086
OS/2 1.xx	80286
OS/2 2.xx	80386

Figure 2. Minimum Processor Requirements

The leading diagonal of yes's is self explanatory (Windows will run Windows programs etc.). The no's are immediately understandable simply by looking at the minimum processor requirements shown in Figure 2.

The "yes" indicating that OS/2 2.xx will support OS/2 1.xx applications reflects the that this has always been a stated goal for OS/2 2.xx. WLO, then, fills the gaps allowing Windows applications to run on OS/2 1.xx and therefore on OS/2 2.xx. The "yes" in the yes/WLO entry reflects IBM statements of intent to support Windows applications directly in OS/2 2.xx or later.

HOW DOES WLO WORK?

The most important thing to understand about WLO is that it does not convert the application — it only allows a Windows application to run under PM. The application running under PM will still be fundamentally an MS Windows application using MS Windows window handles, etc.

WLO exploits the fact that the MS Windows and OS/2, executable file formats are identical thus the OS/2 loader can load an MS Windows program. MS Windows, like OS/2 provides its API in a number of dynamic link libraries. WLO simply provides OS/2 versions of these libraries and thereby effectively emulates Windows on top of OS/2. These libraries:

- KERNEL.DLL
- USER.DLL
- GDI.DLL
- KEYB.DLL
- SOUND.DLL
- SYSTEM.DLL ...

are collectively known as the mapping layer. There are two key observations to be made about the mapping layer

- The mapping layer is an emulation of Windows under OS/2 - the mapping layer does not offer access to OS/2 specific features
- Since the mapping layer is simply a set of OS/2 DLL's, and the Windows program can access these by a simple procedure call, it can also access any other DLL in the system. Every other DLL in the system of course includes the whole OS/2 API so there is no restriction to the addition of OS/2 functionality to the program. In practice it will be necessary to separate the Windows and OS/2 functionality into different source modules to avoid the conflicting definitions in the header files WINDOWS.H and OS2.H.

WLO RESTRICTIONS

Here is a brief list of the restrictions that may be encountered when migrating a Windows program using WLO:

- All system interrupts must be rewritten as equivalent C or Windows 3.0 calls for WLO to operate correctly. This may be a problem when using third party libraries.
- The default fonts supplied by PM are visually quite different from those supplied by Windows. WLO supplies OS/2 versions of the Windows fonts for applications that need to maintain their Windows visual appearance in the PM



Setting up and using WLO is extremely simple

environment. WLO also supplies a font conversion utility CONVFONT.EXE which can be used for the conversion of other fonts.

- Windows global memory allocation functions are mapped to segment allocation functions under OS/2 1.xx. Since segments are a relatively scarce resource under OS/2 1.xx, excessive global memory usage by a Windows program could be a problem.
- Windows and OS/2-PM use different code pages (1004 and 850) so use of code points 128 through 255 (common in multi-lingual applications) will involve additional conversion work.
- Printer and device drivers may be a concern as WLO offers no support for Windows printer or device drivers. However existing PM drivers are available to WLO programs and will appear as Windows drivers.
- OS/2-PM is generally a much more secure environment than Windows. PM programs generally run with minimum privilege (ring 3) whereas Windows programs run with a higher level privilege (ring 1). This may cause some incompatibilities.
- The register state on entry to the initialization routine of a dynamic link library is different under Windows and PM so DLLs need a modified initialization sequence. WLO provides an include file CONVDLL.INC and a utility CONVDLL.EXE to resolve this difference but this does mean that migrating a DLL involves a re-compilation as well as a re-link.
- WLO does not provide sound support.
- WLO does not provide LIM EMS/XMS support, though this is not needed for Windows 3.0 applications. (Note particularly though, EMS availability queries are supported — indicating that EMS is not available. XMS queries are not supported at all.)

- The configuration files, OS2.INI and OS2SYS.INI, are binary files accessible only through the OS/2 'Prf...' API calls. WLO maps calls to 'GetProfileString' etc. that access known system entries in WIN.INI to 'Prf...' calls that access appropriate entries in OS2.INI.

- A major feature of WLO is the "opaqueness" of the mapping layer by which I mean that almost nothing can be meaningfully exchanged between the Windows and OS/2 parts of a WLO program (e.g. window handles, messages, etc.). There are two notable exceptions however: clipboard and DDE.

WLO provides support for communication between Windows and OS/2 programs via these two mechanisms and provides any necessary data translation. For example, a Windows bitmap placed on the clipboard by a Windows program can be pasted by a PM program as a PM bitmap.

USING WLO

Apart from a few environmental considerations, setting up and using WLO is extremely simple. On the workshop machines we currently install the following:

- IBM PC DOS 4.01
- IBM OS/2 SE 1.3
- Microsoft OS/2 1.2 Toolkit
- Microsoft C 6.00A
- Microsoft Windows 3.00
- Microsoft Windows 3.00 Software Developer's Kit
- Microsoft Windows Libraries for OS/2 WLO 0.9
- IBM OS/2 1.3 Toolkit

Although this list changes on a regular basis as new products and upgrades are released, the order of the above list is not arbitrary. This is the order in which the various components are installed on the system and is the reverse order in which components should appear on the various path environment variables (PATH, LIB, LIBPATH, DPATH, INCLUDE, HELP, HELPFILES, etc.).

Given all or part of the above installation, there will be a huge number of duplicate files. This file duplication makes the path variables order-sensitive and creates very long path strings. A technique that we have found useful on the Workshop installations is to install all of the tools packages (not DOS, OS/2 or Windows) over each other in the same directories. Be careful if you decide to try this because the order of installation then becomes critical, but there are a number of advantages:

- The paths are easier to get right because most duplicate files are eliminated
- The paths are shorter
- Many (10+) megabytes of disk space saved
- A platform and target (Windows & OS/2) independent environment can be established

The ability to create such a configuration depends on two key features:

- WLO supplies a new version of the PM resource compiler with the name RCPM.EXE thus avoiding the name clash with the Windows RC.EXE
- The Windows header files are suitable for both Windows and OS/2 development. (That is, they have the appropriate #ifdef _MT sections etc.)

BUILDING WLO APPLICATIONS AND DLL'S

Building a WLO 0.9 application is straightforward:

First, re-link the Windows application using the /NODEFAULTLIBRARYSEARCH switch and use the WLO and OS/2 libraries:

`LIBMK_B.LIB SLIBCEMK.LIB OS2.LIB`

instead of the Windows libraries:

`LIBW.LIB SLIBCEW.LIB`

Second, re-attach the Windows resources to the resulting .EXE file using the Windows resource compiler RC.EXE.

Building a WLO DLL is slightly more complex as it involves a source file change and re-assembly as well as the re-link. The sequence of operations is as follows:

1. Edit the library initialization source to include the WLO file CONVDLL.INC as documented in the WLO manual and reassemble.
2. Re-link the Windows application using the /NODEFAULTLIBRARYSEARCH switch and the WLO and OS/2 libraries:

`LIBMK_B.LIB SDLLCEMK.LIB OS2.LIB`

instead of the Windows libraries

`LIBW.LIB SDLLCEW.LIB`

3. Re-attach the Windows resources (if any) to be added to the resulting DLL using the Windows resource compiler RC.EXE

That's all there is to it. Providing the application does not violate one of the restrictions listed above, it should now be able to run under OS/2. Note that to market the WLO application, developers must ship the retail mapping layer DLL's with the application.





```

## QA Training: Windows & PM Sample program MAKEFILE ##
all: ..\binp\winsamp.exe ..\binp\migsamp.exe ..\binp\pmsamp.exe

## Build the PM icon ##

migsamp.ico: winsamp.ico
convicon -f winsamp.ico temp
copy temp.ico migsamp.ico
del temp.*

## Compile the Windows resources ##

winsamp.res: winsamp.rc winsamp.ico winsamp.cur dlgbox.h
rc /r winsamp.rc

## Compile the PM resources ##

pmsamp.res: pmsamp.rc migsamp.ico pmsamp.ptr dlgbox.h
rcpm /r pmsamp.rc

## Compile the Windows source file ##

winsamp.obj: winsamp.c dlgbox.h demo.h
cl /W4 /Gsw /Zpe /c winsamp.c

## Compile the PM source file ##

pmsamp.obj: pmsamp.c dlgbox.h demo.h
cl /W4 /Gsc /MT /c pmsamp.c

## Link the Windows program ##

winsamp.exe: winsamp.obj winsamp.def
cl /W4 /Gsw /Zpe $** /Fe$@ /link /ALIGN:16 /NOD libw sllibcew

## Link the Migrated program ##

migsamp.exe: winsamp.obj winsamp.def
cl /W4 /Gsw /Zpe $** /Fe$@ /link /A:16 /NOD libmk_b sllibcemk os2

## Link the PM program ##

pmsamp.exe: pmsamp.obj pmsamp.def
cl /W4 /Gsc /MT $** /Fe$@ /link /ALIGN:16

## Attach the compiled Windows resources to the Windows program ##

..\binp\winsamp.exe: winsamp.exe winsamp.res
rc winsamp.res winsamp.exe
copy winsamp.exe $@

## Attach the compiled Windows resources to the Migrated program ##
## and add the PM icon as an extended attribute so the program ##
## will appear correctly when added to a PM group ##

..\binp\migsamp.exe: migsamp.exe winsamp.res migsamp.ico
rc winsamp.res migsamp.exe
wloinst -i migsamp -e migsamp
copy migsamp.exe $@

## Attach the compiled PM resources to the PM program ##

..\binp\pmsamp.exe: pmsamp.exe pmsamp.res
rcpm pmsamp.res pmsamp.exe
copy pmsamp.exe $@

```

Figure 3. MAKEFILE Listing

It is perhaps worth noting here that Microsoft recently announced some information about WIN32 — a future 32-bit version of Windows. Contained in the announcements was a statement that in order to prepare for WIN32, applications should be written such that they migrate correctly with WLO now.

The moral of the story is that if your application does not work under WLO you have discovered a problem with your application, not with WLO! The general rule when using WLO as a migration route is to first fix the problems by changing the Windows source, and only then consider adding OS/2-PM extensions.

Figure 3 shows the MAKEFILE for an example program used in the Workshop. The MAKEFILE builds a native Windows version, a native PM version and a WLO version of the application. This makefile shows clearly the simplicity of building these different types of application with the common environment suggested above and also shows that MAKEFILEs can be created to build Windows and WLO versions of an application simultaneously.

APPLICABILITY OF WLO

Now we know what WLO is, how it works and how to use it. We now need to address one last issue — when to use it.

When only the alpha releases of WLO were available, it was not possible to create a stable commercial product based on it. However, even at that stage WLO was a useful product — it could be used as a platform for incremental migration. The concept of incremental migration is to use WLO for a quick port and then incrementally replace Windows functionality with PM functionality until you have a pure PM application.

At this point in time WLO is a stable product and for applications that are not currently pushing the limits of Windows technology a straight WLO port may be sufficient. Microsoft currently claims that a WLO application will run approximately 15 percent slower than a PM program, but I would suggest that this should not be taken too literally. Fifteen percent may well be the

penalty for mapping Windows functionality onto PM functionality through the mapping layer but real comparisons are difficult to obtain. The penalty for attempting to run an application with Windows structure rather than PM structure probably represents a much higher penalty in many cases.

Nonetheless, WLO is now a stable product and it is certainly reasonable to consider shipping certain products in this form.

At the Migration Workshops we have seen many applications of varying size and complexity. These applications have been created by companies with widely differing resources and experience. We cover WLO as just one item in the arsenal of weapons to attack the migration problem. The sections below describe some of the thinking of previous participants at the Workshop.

THE BENEFITS OF WLO

The benefits of WLO are self evident — for a few minutes work with WLO it is possible to create a PM version of most Windows applications with equivalent functionality. This represents a very cost effective means of extending the potential customer base for many applications.

Even if WLO represents the limits of investment in OS/2 technology for many companies, this should help to solve one of OS/2's major problems — the dearth of PM applications.

Company strategies for approaching the Windows-to-OS/2 migration problem differ widely. There seem to be two basic reasons for the difference:

- **Motivation:** some companies are motivated by the desire to extend the customer base for an application that exists comfortably in the Windows environment. Others see OS/2 as the solution to Windows restrictions on connectivity, performance, memory,etc.
- **Resources:** some companies with limited resources cannot justify shifting resources from Windows development. Others see OS/2 development as essential and will therefore make resources available as required.



If your application doesn't work under WLO, the problem is with your application not WLO!



Combinations of these variables explain the various attitudes to OS/2 migration seen at the Workshops:

Motivation: Extend customer base
Resources: Limited.
 Companies in this position typically see WLO as a complete solution. Future strategies are limited to developing all new Windows applications within the WLO constraints.

Motivation: Extend customer base.
Resources: Available.
 For companies in this position, WLO provides a complete solution, but the availability of resources allows the development of PM specific extensions. Future strategies cater to the parallel development of OS/2 and Windows products.

Motivation: Need of OS/2 functionality.
Resources: Limited.
 Companies in this position typically adopt the hybrid approach. WLO solves no problems on its own, but the addition of PM functionality to a WLO application can provide some of the required functionality without the major overhead of a total rewrite. Future strategies cater to the parallel development of OS/2 and Windows products.

Motivation: Need of OS/2 functionality.
Resources: Available.
 For companies in this position WLO probably doesn't hold very much interest. The best product will be achieved by native OS/2 PM development. Future strategies cater to the parallel development of OS/2 and Windows products.

SUMMARY

WLO offers a simple and convenient way of getting Windows applications running under OS/2. WLO is just one of a number of strategies we offer at the Windows Migration Workshops (see notes). The usefulness of WLO depends on the nature of the application and the motivation for the migration. WLO is an excellent product but it is not a magic potion with the solution to all migration problems.

For more information about the Developer Assistance Workshops, call (407) 982-6408.

Phil Spencer, QA Training, Ltd., Cecily Hill Castle, Cirencester GL7 2EF UK Tel: 011-44-285-655888 Fax: 011-44-285-650537. QA Training offers world-wide training and consultancy in OS/2 Presentation Manager, DOS, MS Windows, Languages, Unix, etc. Mr. Spencer holds a B.Sc. in Aeronautical Engineering and an M.Sc. in Computing Science, both from Imperial College, London. He has a long history of developing real-time microprocessor-based operating systems before joining QA in 1987 where he now specializes in OS/2-PM training and consultancy. He can be reached directly on CompuServe 100014,104.

Software Tools

Porting Real-World Windows Applications to OS/2 PM: In Search of a Total Solution

by Richard Merrick, Kurt Delimon, Wes Bell and Ron Howell

*During the past 10 years, Micrografx has played a key role in establishing Microsoft Windows™ as an accepted operating environment. Beginning in 1982 with the release of PC Draw™ and later Windows Draw!™ and In*A*Vision™, the Micrografx product line has evolved to include both Windows and OS/2 Presentation Manager™ versions of Micrografx Designer™ and Micrografx Charisma™. Designer provides both desktop publishing and advanced technical illustration features, while Charisma complements Designer with data-driven business graphics and word charts for composing sophisticated presentation graphics.*

In this article, the author discusses Micrografx's experience with OS/2 conversions, and the company's own conversion tool, Mirrors™, which is currently licensed to IBM and is expected to be available to other software developers early next year.

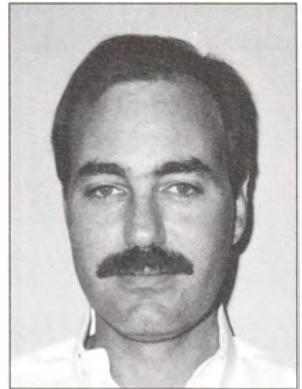
In the fall of 1988, we at Micrografx began to realize the considerable amount of effort required to port all of the components of our existing applications to OS/2 Presentation Manager (PM). We planned on porting two major applications which were already shipping, others that were on the way, as well as the periodic updates. At an estimated six months per port and \$100,000 per application, we decided to explore the possibility of developing emulation libraries which would enable our Windows applications to run under PM. This initial research culminated in the development of Micrografx Mirrors, a porting tool used to port Windows applications to OS/2 PM. Our company objectives for our OS/2 PM applications were ambitious from the very beginning:

- Release the same version of software at virtually the same time in both Windows and OS/2 PM.

- Maintain identical functionality and behavior between Windows and PM versions wherever possible.
- Maintain near or better performance of the PM version as compared to the Windows version.

The only way to attain these goals was to write a general-purpose porting layer that would work consistently with subsequent releases of our applications so creation of the PM versions could proceed in parallel with our Windows development cycles. Because our applications rely heavily on the Windows device driver architecture for both printer device support and for bitmap management and metafile creation, we had to find a way to port Windows device drivers. If this were not enough, we realized that performance under PM must be at least as good as under Windows, or there would be no incentive to purchase our products.

The Micrografx Mirrors™ technology was licensed to Microsoft in early 1990 and was the precursor of the Microsoft Windows Libraries for OS/2, or WLO. While Microsoft pursued libraries which could be useful as both a development toolkit and binary compatibility layer, we pursued Mirrors as a high-performance porting tool intended for use only as a development toolkit. Recently, IBM and Micrografx announced a joint development agreement whereby Mirrors, and related technology, is to be extended to take advantage of the new OS/2 32-bit architecture with the intention to provide this to independent software vendors. The performance gains of flat-model addressing, combined with many other static and dynamic optimizations, promises to yield performance of a Windows application ported to OS/2 PM equal to or better than the same application under Windows 3.0.



Richard Merrick

Mirrors is being extended to take advantage of OS/2 32-bit architecture

PORTING APPLICATIONS WITH MIRRORS

Today, porting our Windows application Micrografx Designer to PM has become an integral part of our development process, to the extent that the same version 3.1 currently

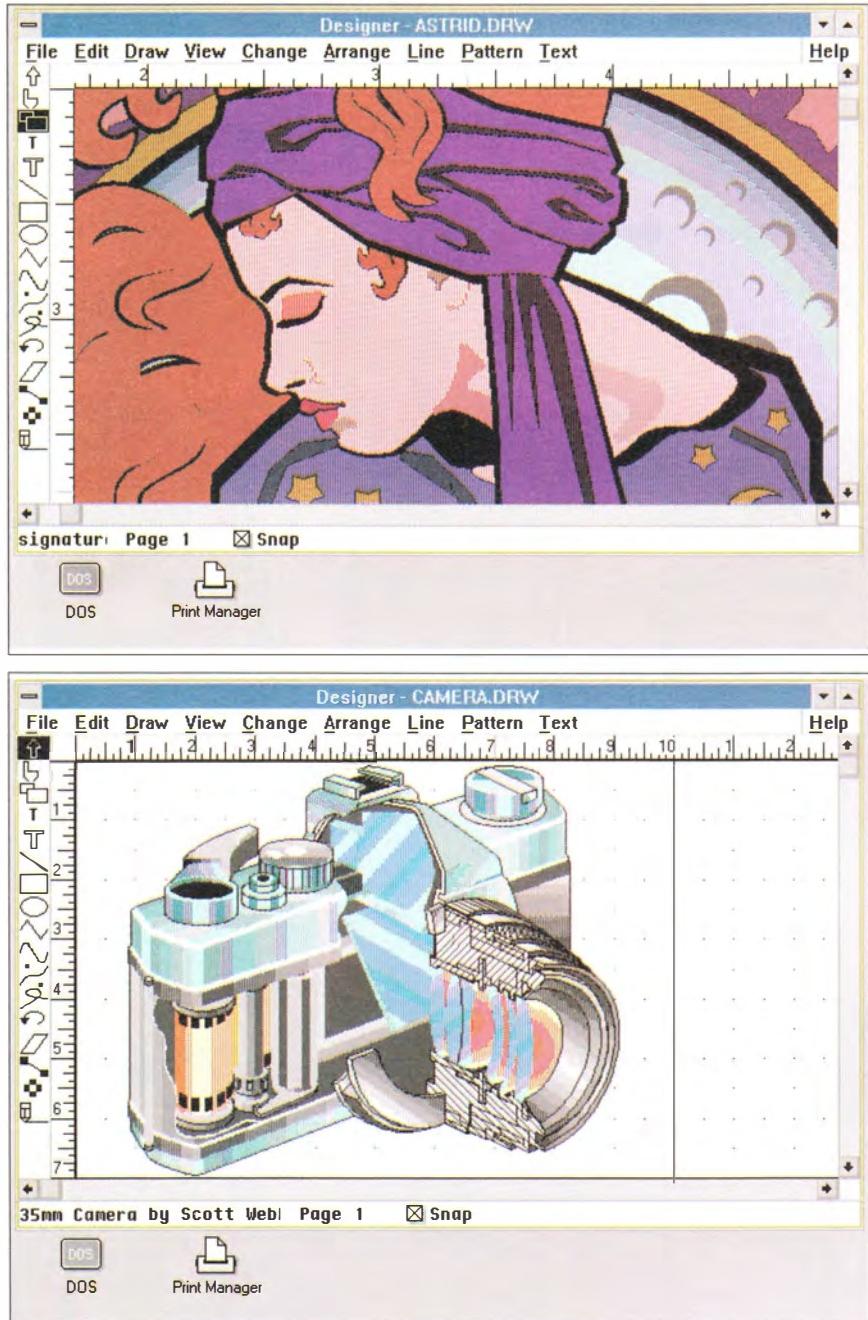


Figure 1. PM Designer 3.1 Using Micrografx Mirrors

offered under Windows is provided for PM (see Figure 1). Designer, first ported with Mirrors as a Windows 2.0 application, has become much easier to port since the release of Windows 3.0. This is mostly due to the restrictions imposed by running in protected mode, also required for execution in PM, which enables parallel development in both Windows and PM.

The first step in porting Designer using Mirrors involved the elimination of DOS dependencies by using functions provided in the Mirrors MHOST library. MHOST.DLL contains functions for replacing DOS INT21 calls, such as file I/O, date and time, and path and directory navigation, and uses the same API for both DOS and OS/2. Designer is then linked with the Mirrors emulation libraries, MUSER, MGDI, and MKERNEL, in place of the Windows libraries. These libraries export all functions necessary to satisfy most Windows API bindings. As part of the build process, the Mirrors MIRRC and MIRDEF utilities are used to convert Windows resources, such as bitmaps, icons, and cursors, and the Windows module definition file into PM compatible formats. Windows Help, in the form of rich text (RTF) is also converted by the MIRHLP utility into the PM IPF help format. When the application runs under PM, the Mirrors emulation libraries are loaded, and they map Windows calls to PM calls while performing message filtering and reordering. Figure 2 illustrates several applications ported from Windows to PM using Mirrors.

With widespread interest in Mirrors growing during Spring, 1989, we decided to make it available as a toolkit to other software developers. We beta-tested Mirrors with a variety of applications and immediately encountered a number of challenging problems, including undocumented Windows functions, differences between the Windows and PM message models, and use of non-standard memory management techniques.

MESSAGE MODEL DIFFERENCES

Without a doubt, our greatest challenge with Mirrors was in correctly mapping the Windows message model onto PM. For instance, a typical Windows application calls CreateWindow() followed by UpdateWindow() before it enters its message processing loop. This results in an application's window procedure receiving a WM_ERASEBKGRND message prior to receiving a WM_SIZE message. A PM application, on the other hand, will receive the same messages in the reverse order.

Another difference between the Windows and PM message model is the method of erasing and painting the client background. In Windows, an application can do all of its background painting by processing the WM_ERASEBKGRND message. It may draw in or fill the client area with any color or pattern. It could, for example, fill the background with a light gray brush and then draw a grid with a white pen. It may also pass this message to DefWindowProc(), which will fill the invalid portion of the client with the brush registered for this window class. If there is no brush, then the background is not filled. This can be done because an application will always receive a WM_ERASEBKGRND message prior to painting its client area. This message may be sent by Windows prior to the WM_PAINT message or as a result of the application calling the BeginPaint() function. This behavior in Windows holds true for all window styles.

In OS/2 PM, all painting, including erasing the background, should be done when an application receives a WM_PAINT message. An application cannot count on receiving a WM_ERASEBKGRND when a portion of its client area becomes invalid. This is because the frame window sends the message to the client's window procedure before it paints itself. If the frame window is not invalidated or the client does not have a frame window it will only receive a WM_PAINT.

To duplicate this Windows behavior, Mirrors sends the application's window procedure a WM_ERASEBKGRND when a portion of the client becomes invalid.

This sounds easy enough, but the fact that the wParam of the erase message contains a handle to a Windows device context (DC) complicates the process. Because the application can use this DC handle to paint its client, a Presentation Space (PS) handle must be acquired. In addition, the clip region of this PS must be set to the update region of the invalid portion of the client.



MEMORY MANAGEMENT DIFFERENCES

Memory management is an area where some undocumented features of Windows caused problems in the early versions of Mirrors. According to the Windows documentation, the correct use of global memory is to first allocate it then lock it to get a pointer to the memory. In an effort to improve performance, some of our first Mirrors customers relied on the fact that when memory was allocated fixed, the memory handle was actually the selector of the allocated memory block. By casting this handle to a pointer with an offset of 0 bytes, a call to GlobalLock was avoided. Mirrors does not support this behavior because it is entirely device dependent and could not be supported in the flat memory model of OS/2 2.0.

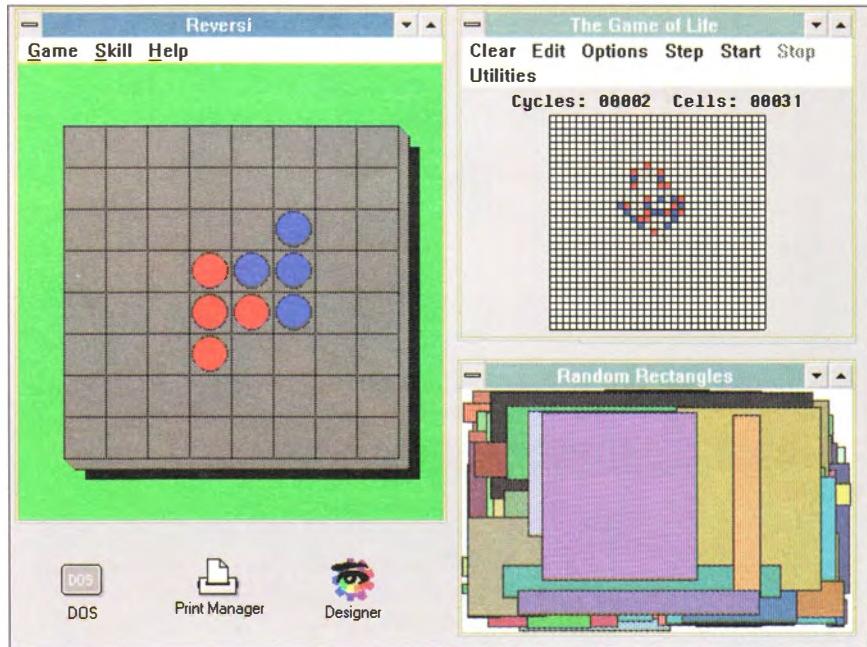


Figure 2. Multiple Applications Running Under PM Using Mirrors



IMPROVING THE PERFORMANCE OF MIRRORS

In order to achieve our performance goals, we made several important design decisions.

- Perform static conversion of resources.
- Implement an intelligent mapping layer which exploits opportunities to cache information wherever possible.
- Avoid adding a layer wherever possible by implementing our own control classes.

Static Conversion of Resources

Binary Windows resources (such as bitmaps, cursors, and icons, referenced in a Windows resource file) are converted from Windows format to PM format whenever the Mirrors MIRRC resource file converter is run against the Windows RC file. This improves the overall application performance by loading the resource as a native PM resource without having to go through a conversion process during run time. Note that this only applies to resources that are loaded from the resource file. Resources that are created at run time by the application must still go through conversion from Windows to PM format, and vice versa, at run time. In the same way RTF files, used to create Windows help, are converted into PM IPF files for direct use by the PM help facility. This not only improves performance, it provides consistency with native PM applications.

Caching and Performance Tuning

In the process of porting applications with Mirrors, we profiled the internal and exported Mirrors functions to determine which functions are being called most often and how we might improve their performance. One of the most significant performance increases has been in the area of object selection. A Windows application will typically select an object, such as a red pen, draw with that object, and then select the previous object back into the DC. In Mirrors, a SelectObject call generates a GpiSetAttrs() call to PM to set the appropriate attribute in the PS. For a drawing application, such as Micrografx Designer, the number of SelectObject() calls made can be very high. Many times the object will be selected into the DC, used to draw, and selected out of the DC only to be selected immediately back into the DC to draw again.

In order to optimize this situation, Mirrors does not actually realize an object, or call GpiSetAttrs(), to change an attribute with PM until the object is needed to perform some type of output. By doing this, Mirrors will only realize the objects that are actually needed to perform output, and many times an object will remain realized with PM to perform several output functions before being unrealized. Designer can actually draw some pictures faster using Mirrors and PM than in Windows because of this intelligent object selection and other optimizations. Recent benchmarks show complex drawing performance as much as 8% to 14% faster in OS/2 PM 1.3 than under Windows 3.0. Of course, the optimization made by this type of realization is very application dependent. However, in the very worst case, Mirrors will do no more than a one-to-one realization of selected objects.

Another Mirrors feature that increases performance is allocation of a block of memory with the DosAllocSeg() function and then sub-allocation out of that block with the DosSubAlloc() function. We found that many times we were calling DosAllocSeg() to allocate a very small block of memory, which can be very expensive. By using DosSubAlloc(), we were able to allocate small blocks of memory for internal use without much overhead.

Yet another optimization in Mirrors is in the handling of device driver queries. Some applications make numerous calls to the screen device driver to get information. Many times, this same information will be queried over and over. Mirrors optimizes this by querying and saving the static screen driver information. Whenever a call is made to query the screen driver information, Mirrors simply returns the information without making a call to the driver. In addition to revealing areas of inefficiencies, Mirrors also helps reveal errors we might not have otherwise found. At times, our applications will attempt to do something that is invalid, such as select a NULL handle into a DC. By profiling the SelectObject() function in Mirrors, we can determine when this is happening and provide a fix for the application. We have found situations where applications will overwrite memory and cause a GP fault under PM, which did not happen in Windows, even in protected mode.

Mirrors Control Classes

While PM provides the same control window classes as Windows, all of the default Window classes such as button, listbox, and scrollbar are now part of Mirrors. PM controls were used initially and later abandoned in order to improve performance and correct differences in the message models. Because PM controls were used prior to the shipment of Windows 3.0 and Windows 2.x used a monospaced font, Mirrors selected a monospaced font in the PM controls by calling the function WinSetPresParam(). This achieved the desired effect, but only at the expense of performance. In fact, it takes twice as long for a dialog box to display when a font other than the system default is used. Even though PM controls perform admirably in native PM applications, it became clear that we could not achieve our performance goals for Mirrors by using the PM controls.

The other problem in using PM controls was incorrect behavior when compared with their Windows counterparts. For example, in Windows you can control the appearance of a child control window by processing the WM_CTLCOLOR messages that they send to their parents prior to painting. This gives an application the ability to perform font selection, text foreground and background colors, and background brush used by the control window when it paints. The only way to achieve the same effect while using the default PM controls is to subclass each control and, when the WM_PAINT message is received, send the parent window a WM_CTLCOLOR message. The wParam of this message should contain a Windows DC handle that the parent window uses to select fonts, set text colors, etc. This requires getting a PS, building a Mirrors (Windows) device context structure and setting up a clipping region in this PS — all of which have a negative impact on performance.

Moreover, this does not handle the mapping of Windows API calls, such as SetTextColor(), into the appropriate values in the WinSetPresParam() function.

These performance and functional limitations caused us to implement all of the default classes in Mirrors. This allowed us full control over the behavior and appearance of controls in a ported application. The performance was also improved over that of the PM classes when used as part of the porting layer.

PORTING WINDOWS DEVICE DRIVERS WITH OASIS



Micrografx has historically provided high-quality printer device support to complement our applications. For OS/2 PM, our device driver strategy is based on porting our Windows device drivers to PM using an emulation layer similar to Mirrors. This device driver porting layer, dubbed Oasis™ has the primary requirement of providing presentation drivers which work transparently with both native PM and ported Windows applications, while offering the same functionality under PM that they have under Windows.

As demonstrated in Figure 3, the Oasis architecture consists of three components.

- The Windows device drivers as an OS/2 dynamic link library.
- The OS/2 PM presentation driver instance for a given Windows device driver.
- The Oasis dynamic link library (DLL).

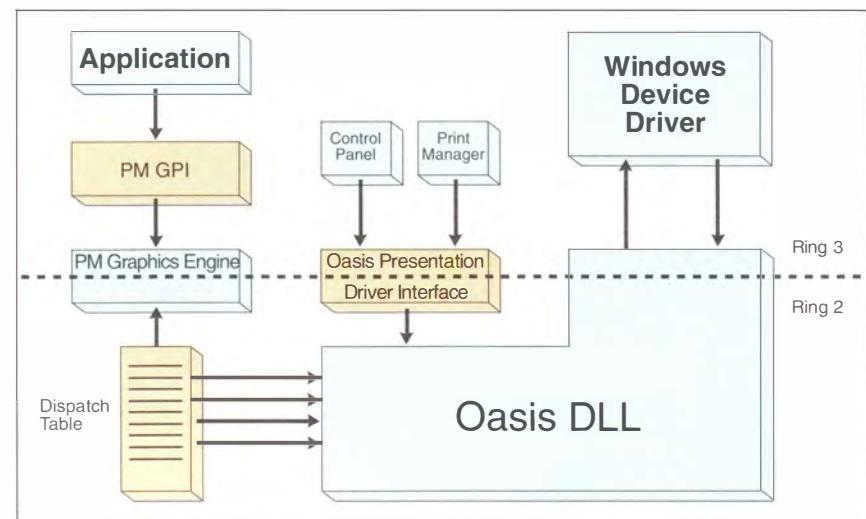


Figure 3. Oasis Presentation Driver Architecture

Windows device drivers are first made host-independent by removing DOS dependencies via conditional compiles for operating system specific calls or use of the Mirrors MHOST library. For our use, we used MHOST to provide platform independent I/O and directory services. The next step is to create a MAKE file for each Windows device driver. This MAKE file can be created dynamically by writing a C source code file that is compiled using the C pre-

processor and conditional compiler directives. During the MAKE process, the Windows device driver is linked against Mirrors, or Microsoft WLO and a surrogate PM presentation driver is created. A PM presentation driver skeleton is provided with Oasis to device driver developers, for use in installing the Windows device driver into the Print Manager.

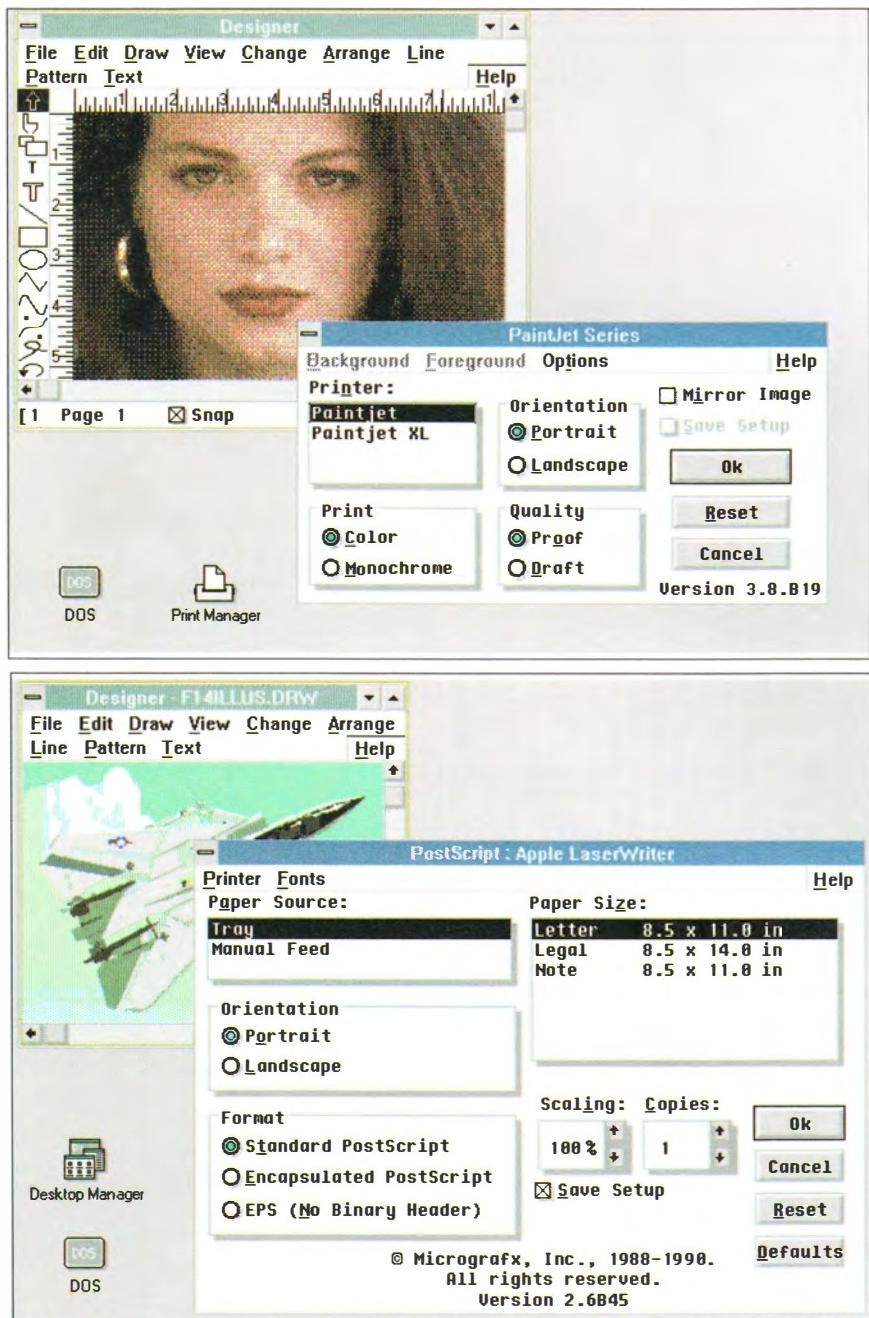


Figure 4. Micrografx's PaintJet and Postscript Drivers for PM using Oasis

In Figure 4, the Micrografx Windows PostScript Driver and HP PaintJet Driver dialog boxes are presented running under OS/2 PM 1.3 using Oasis. We have already ported other Micrografx Windows device drivers, including the Micrografx plotter driver (supporting IBM, Bruning, and Roland plotters), and Matrix film recorder drivers.

Early on, even though we were certain that a driver porting layer was possible, we were concerned about some of the unknowns, such as how would the driver install into the PM Print Manager, how would we implement the PM Spooler interface, what about passing printer escape support down, or how would device fonts be handled? The project turned out to be much more manageable than we expected, thanks to prior experience hooking out the PM graphics engine.

HOOKING THE PM GRAPHICS ENGINE

Unlike the Windows Graphics Device Interface (GDI) to Windows device driver interface, the OS/2 PM Graphics Rendering Engine (GRE) has only one entry in a presentation driver that it invokes. This exported entry is named OS2_PM_DRV_ENABLE(). This entry is used by the GRE to enable a DC, disable a DC, reset a DC, and to save and restore the state of a DC.

During the enabling of a DC in OS/2 PM, GRE makes four calls to OS2_PM_DRV_ENABLE() to complete the enable process. When GRE makes the first of the four enable calls to a given presentation driver, GRE passes a pointer to an array of long pointers to graphics functions. This array of long pointers to graphics functions is known as the dispatch table. Initially, all of the function pointers in the dispatch table point to functions in GRE (PMGRE.DLL). The dispatch table may contain up to 256 pointers, 149 of which are currently used in OS/2, version 1.3.

Every presentation driver must hook out 73 of these pointers, and may hook out any of the remaining 76 pointers as is required by the particular presentation driver. To hook out one of the pointers is to update the dispatch table at a particular array index to point to a graphics handling function in the particular presentation driver. OASIS.DLL handles this process on behalf of the surrogate PM presentation driver by hooking out over 95 of the graphics function pointers.

In Windows, device drivers run at Ring 3, while in OS/2 1.x presentation drivers run at Ring 2. One of the first design considerations was how to hook GRE as a presentation driver running at Ring 2, while resolving the Windows' driver interface with Mirrors or the Microsoft WLO at Ring 3. If the code for the Windows device driver was linked to run at Ring 2, then Windows API calls (which are resolved by Mirrors) would have to be made through some mechanism, such as going through function DosCallback() to make the ring transition from Ring 2 to Ring 3 for every Windows API call that was made by the Windows device driver.

We decided that the Windows device drivers would be linked to run at Ring 3 as OS/2 dynamic link libraries, allowing Oasis to have almost all of its code segments be Ring 2 code segments as required by an OS/2 presentation driver. The ring transition is then handled inside Oasis which contains a Ring 3 code segment that is called via function DosCallback(). This Ring 3 code segment loads the Windows device driver during the enable process, and obtains the procedure address for all required Windows device driver EXPORTs. These EXPORTs are then called directly from Oasis at the Ring 3 peer level.

WINDOWS DEVICE DRIVER ESCAPES

Another hurdle to overcome involved support for Windows device driver escapes. Windows and PM drivers often provide special time saving capabilities through an exported Escape() entry point. OS/2 PM defines only 14 printer escape codes. Windows on the other hand defines over 60 printer escape codes, of which only 10 map directly to OS/2 PM printer escape codes. The additionally defined Windows printer escape codes are mapped to one of the OS/2 PM user definable ranges by Mirrors, and then Oasis maps these additional escape codes back to their Windows values before calling the Windows device driver. Even though the Microsoft WLO does not currently support mapping of these additional escapes, Windows device drivers ported with Oasis will still operate with some reduction in performance.

SUPPORTING THE PM SPOOLER

When it came to implementation of PM Spooler support in Oasis, we confronted some distinct differences between the Windows spooler and the OS/2 PM spooler. The first difference is the driver interface (or API) to these spoolers. The second difference is functionality provided by these different spoolers. A third difference is the manner in which the spooler output is sent to the destination printer port. In order to handle both ported Windows applications and native PM applications, three different emulation modes were implemented in Oasis to handle all cases of output that may be generated by a printer driver.

The three emulation modes allow for output directly to a disk file, output directly to a physical port, and output to the PM spooler in either PM_Q_STD or PM_Q_RAW format. This last mode presented a problem for PM_Q_STD mode where a PM metafile is created by the PM Spooler for subsequent playback. During this process, some PM escapes — such as DEVESC_NEXTBAND — are not recorded in the metafile during spooling, so certain driver escapes were being lost which were required by the Windows device driver for correct behavior. For example, dropping the DEVESC_NEXTBAND escape causes the Micrografx PostScript Driver to not print the page. The solution used by Oasis is to use



*The OS/2 2.0
32-bit flat
memory model
offers many
opportunities
to improve
performance*

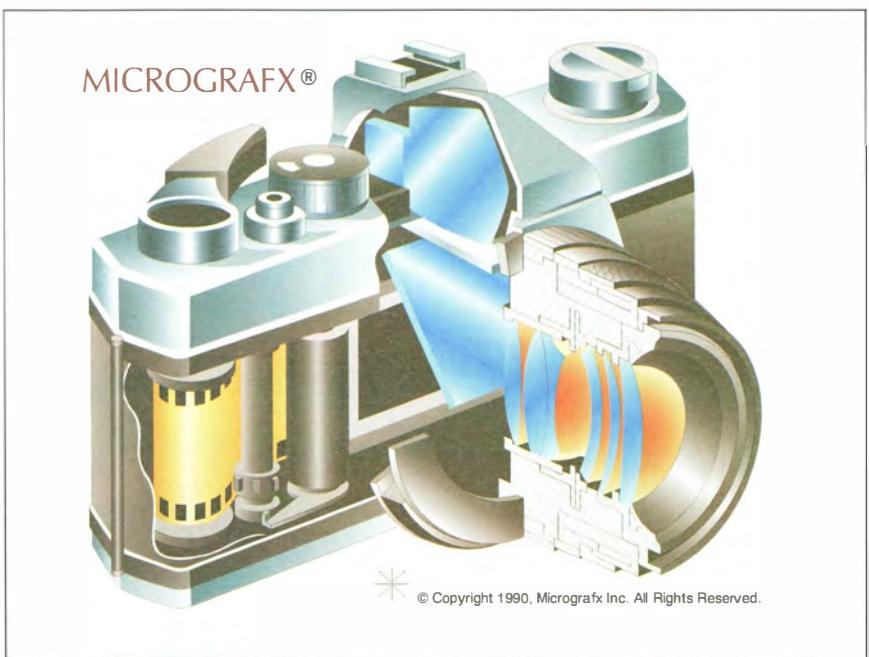


Figure 5. Camera.DRW Printed Using Micrografx PostScript Driver Version 2.6



Figure 6. *Astrid.DRW* Printed Using Micrografx PostScript Driver Version 2.6

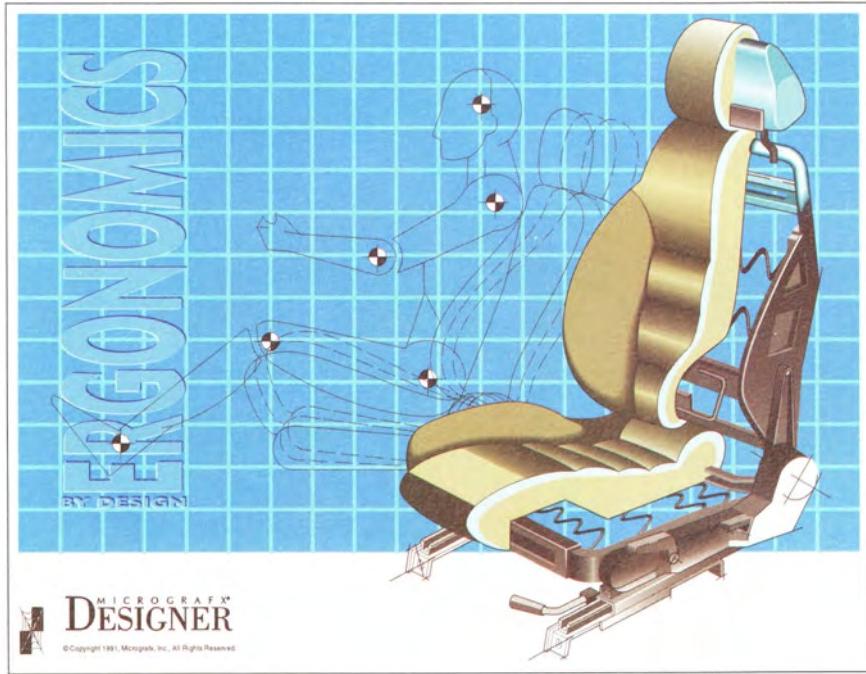


Figure 7. *Carseat.DRW* Printed Using Micrografx PostScript Driver Version 2.6

PM_Q_STD only for vectorplotters and PM_Q_RAW for all other devices.

DELAYED REALIZATION AND UNREALIZATION OF OBJECTS

Whenever the OS/2 PM LINEBUNDLE color or style (for example) are changed, Oasis does not call the Windows device driver function `RealizeObject()` to realize the associated Windows logical pen. Instead, function `RealizeObject()` in the device driver is only called when output to the device driver is about to occur. Therefore, LINEBUNDLE attributes (for example) can be updated by an application for the associated device context, but Oasis will only update the LINEBUNDLE structure for the given device context without requesting the Windows device driver to realize a new pen, thus possibly avoiding unnecessary calls to the Windows device driver.

In Windows, whenever an application selects a new pen (for example), the pen is realized when GDI calls the Windows device driver function `RealizeObject()`. Whenever another pen is selected for a given device context by the application, GDI will call `RealizeObject()` three times in the device driver. The first call will tell the device driver to delete the previously realized pen. The second call will request the size in bytes for the new pen that is to be realized. After GDI allocates the required memory for the physical pen, function `RealizeObject()` is called a third time to let the device driver fill in the device driver defined physical pen data structure. As an optimization, Oasis will cache more than one realized pen (for example), recording the logical pen data structure and the number of bytes that Oasis allocated for the associated physical pen. When a cached logical pen is to be realized, Oasis need not call the device driver to do any work for the new logical pen because Oasis already has a pointer to the associated physical pen.

In Figures 5, 6, and 7, color output from the Micrografx PostScript Driver, version 2.6, was printed on an OceColor PostScript printer under PM using Micrografx Oasis and Mirrors. In fact, Micrografx Designer PM, version 3.1, ported with Mirrors, was used to create the drawing and initiate printing. By providing near identical PM versions of Designer, accompanying utilities, bitmap filters (device drivers), and our HP PaintJet and PostScript drivers simultaneous to the

shipping Windows versions, we were able to achieve our objectives of providing a consistent user interface and functionality for both Windows and OS/2 PM products.

FUTURE DIRECTIONS

The upcoming OS/2 32-bit flat memory model offers many opportunities to further improve the performance of Windows applications ported with Mirrors and Oasis. For instance, Mirrors can take further advantage of flat memory management for memory allocations and various utility functions without surfacing a 32-bit API to the client Windows application. This could have the result of helping ported 16-bit Windows applications go even faster under PM than under Windows. As part of the joint IBM/Micrografx development agreement, Micrografx is working with a team in Boca Raton, Florida, to rewrite the PM graphics engine for the 32-bit architecture. Performance improvements here, coupled with additional functions supporting Windows behavior in Mirrors, should result in dramatic improvements in Windows fonts, dialog boxes, and polygon rendering speed.

Ported Windows device drivers could also be updated to use the flat memory model, instead of the segmented memory model. This should improve performance for the use of large memory allocations because huge pointers, and their associated overhead, will no longer be necessary.

When the OS/2 32-bit graphics engine is updated to the flat memory model, and a given Windows device driver is updated to use the flat memory model, then Oasis could take advantage of this by passing data that is larger than a 64K segment directly to the Windows device driver. This would avoid having to break the data into separate segments resulting in more than one call to the device driver for a given call from Oasis.

Richard Merrick, 1303 Arapaho, Richardson TX 75081, is Micrografx's director of systems development. He previously held the position of R&D manager at CompuTrac, Inc. where he directed development of distributed information management systems. Mr. Merrick holds a BA and MS in Computer Science from the University of Texas at Dallas.

Kurt Delimon, 1303 Arapaho, Richardson TX 75081, is the lead software engineer for Micrografx Mirrors. Prior to joining Micrografx, he held a Software Engineer position at Bell Northern Research where he helped develop a porting layer to run a GEM network management system under Windows. Mr. Delimon holds a BS in Computer Science from the University of North Texas.



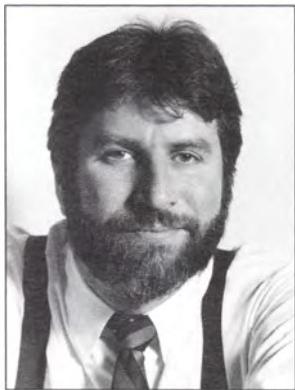
Ron Howell, Wes Bell and Kurt Delimon

Ron Howell, 1303 Arapaho, Richardson TX 75081, is a software engineer working on the Mirrors project and also developed a Windows printer driver for the Hewlett-Packard PaintJet and PaintJet XL. Mr. Howell earned an Associate of Science degree at Paris Junior College and a BS in Computer Science from the University of Texas at Tyler.

Wes Bell, 1303 Arapaho, Richardson TX 75081, is the lead software engineer for Micrografx Oasis. Previously he wrote the OS/2 Plot Queue Processor delivered with OS/2 v.1.3. Mr. Bell is a Summa Cum Laude graduate of the University of Texas at Dallas, with a BS Mathematical Sciences and a major in Computer Science.

Software Tools

Developing PM Applications with Gpf



Michael Drapkin

by Michael Drapkin

Gpf (GUI Programming Facility) from Microformatic U.S.A. Inc. presents the developer with a robust CASE environment for building OS/2 Presentation Manager based applications. GPF cleanly manages and generates SQL, Help, Resource Compiler, Make and C language statements. In addition to its interactive graphical user interface used for creating windows controls, Gpf includes a powerful Animate feature for testing applications without the need for compilation.

A common complaint heard among newcomers to message-based graphical user interface programming environments is about the amount of time and lines of code it takes to perform a simple task, such as the famous "Hello, world." described at the

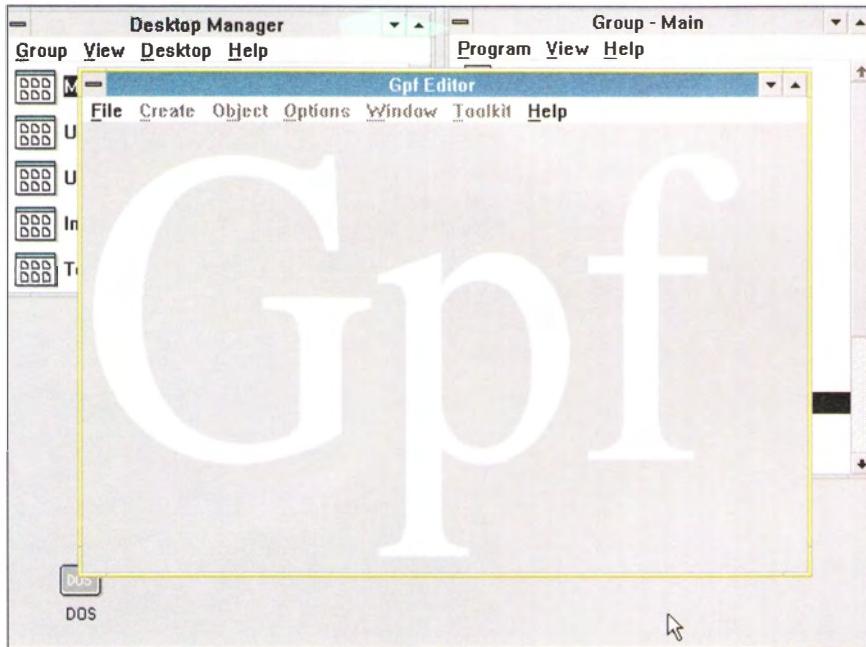


Figure 1. The Main Gpf Menu and WIndow

beginning of Kernighan & Ritchie's *The C Programming Language* book. Many veteran programmers will circumvent this difficulty by creating a skeleton — a generic body of code that they can use as a kind of template for creating the framework of a new application. Similarly, it is common practice to merge snippets of working code from previous projects into a new application because this saves them from having to reinvent the wheel, and worse, to debug it.

Although these techniques work admirably, difficulties arise out of the fact that Presentation Manager based applications usually change, often and sometimes drastically, during the course of a development cycle. This leaves the programmer with the tedious and repetitive task of having to make small changes to a program, then compiling and running the application to see if their changes resulted in the desired effect.

A French programmer by the name of Jean Bernard Clerin decided that there had to be an easier way to manage the development of the user interface aspects of applications under OS/2 PM, so he took a year, went underground, and emerged with Gpf.

OVERVIEW

A simple and effective tool, Gpf allows developers to quickly build applications by managing the creation of code (currently only in the C language) for all of the windows controls that are currently available. In addition, Gpf includes facilities for adding certain SQL calls, as well as full Help support. By eliminating most of the time spent on PM controls, SQL, and Help, GPF lets developers spend most of their time working on the actual application.

THE APPLICATION DEFINITION

Gpf holds all information about the application being created or modified in what is called an Application Definition. When the information is saved, it all goes into a single file of proprietary format. This file can then be saved, opened, renamed and modified using the File pull-down.

The application definition is the only file created and therefore the only proprietary item required in the application building process. The generated files require no special runtimes, DLLs or libraries. In fact, once you generate code with Gpf, you need never use Gpf's facilities again. Every bit of code, therefore, can be completely modified. This is certainly quite a change from many other products available on the market.

GPF DEVELOPMENT

Creating an application with Gpf is accomplished through the features available on the Gpf menu bar (shown in Figure 1) and their underlying windows. In fact, the only function of the main window is to provide the programmer with the pull-down menu functions. In practice, it is often useful to resize the main window so that only the title and menu bars are showing.

STARTING A NEW APPLICATION

The File New action from the Gpf menu bar will start you off creating a new application. After designating the file name and directory where your new application definition will be stored, Gpf pops up a window (shown in Figure 2) where you define a number of properties for the program you are about to create. These include Main Source Name, Level, Copyright, Tasklist Title, Language and Database Name.

The Main Source Name is the eight character file name that will be used for all of the files that Gpf will generate for you later on. These consist of C, SQC (SQL embedded source code), RC, MAK, IDS (constant definitions), CMD (compiler call batch file) and IPF (help panel text).

Level allows you to track the revision number of the application, such as "0.35B" or "1.1." This is a straight character field.

Copyright allows you to insert copyright information, such as "(C) Copyright 1991, The Foo Company," which is entered into all files generated by Gpf.

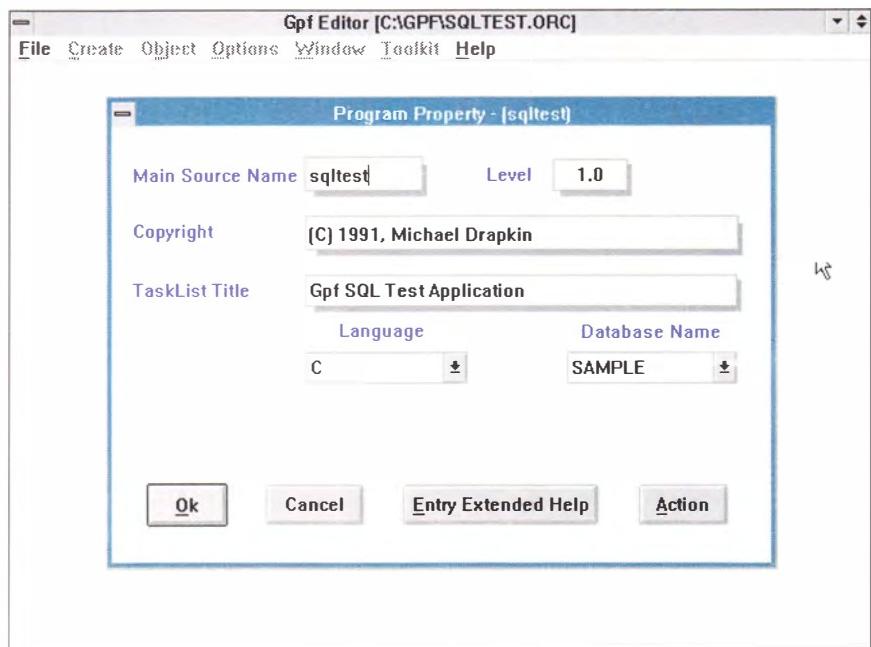


Figure 2. The Gpf Window Defining Various Program Properties

Tasklist Title allows you to specify the title that will appear when the PM task list window is summoned while your application is executing.

Language allows you to specify which programming language you want Gpf to generate code in, although the only choice offered right now is C. Perhaps in the future it will also include Pascal or even the ubiquitous COBOL.

Database Name lets you select an SQL database for use in your application definition. This feature is described later in the article.



PM CONTROLS

The Create pull-down from the Gpf menu bar allows you not only to create windows and dialog boxes, but to add virtually every PM control available. Adding, moving and modifying these controls are very intuitive and easy; to add a control, select the control type from the Create pull-down. The mouse



Figure 3. Setting Control Characteristics

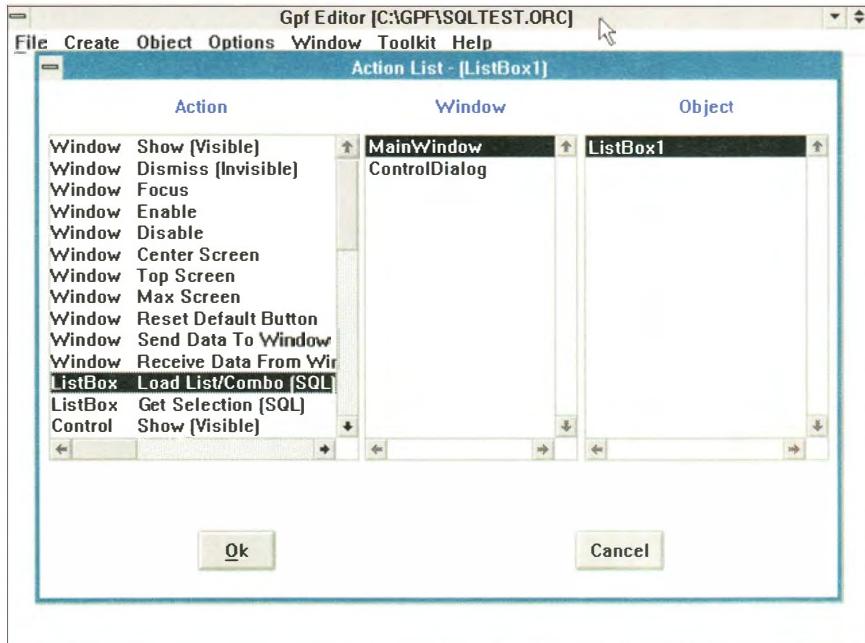


Figure 4. Defining Actions

pointer will change to an iconic representation of the kind of control you are placing. A single click sets it onto your application window, and a window will pop up (shown in Figure 3) where you can modify every conceivable characteristic of the control. Controls can then be dragged for resizing and repositioning. Double clicking will bring up the pop-up window again where you can change the characteristics.

ACTIONS

One of the interesting characteristics that can be set for a control is an Action. Through a series of hierarchical dialog windows and lists, you can select a series of activities to occur. For example, you can specify that when a window is created an SQL table will be loaded into a list box or combo box. Or you can set up a push-button to pop up another window when pressed.

This area is still sparse in terms of the kinds of actions you can specify, and you are limited to the types of actions (shown in Figure 4) that Mr. Clerin has dreamed up for your use. This is sure to be beefed up in future releases of the product. Perhaps he will eventually come up with a way for you to define your own actions that could be added to the pre-defined list.

OBJECTS

What product would be complete today without the current craze: objects. Gpf is certainly no exception, although it puts them to fairly appropriate use. One of the pull-downs on the Gpf menu bar is Object.

Gpf allows you to create four types of Objects: Message Box Objects, Presentation Objects, Icon/Bitmap/Pointers Objects, and User Function Objects.

Message Box Objects allow you to create simple message box windows. During the process of defining a message box object, you may specify the text you want to appear, which system icon (Information, Question, Warning, or Error) you want included, type of alarm (Note, Warning, or Error) and reply

buttons such as OK and Cancel. You can also define characteristics such as default button, modality, as well as whether you want a help button or a title bar (for movability).

A Message Box Object is usually invoked as the object of a Send Text Message Action.

Presentation Objects are some of the more interesting and flexible of the Gpf object types. In a presentation object, you can define any combination of font, foreground, background, or border color (as shown in Figure 5), then apply it to any type of control. Those presentation parameters will then be applied as characteristics of that control. One particularly nice feature of the object-oriented approach to presentation parameters is that when you make a change to a presentation object, all controls that use that object will then assume the new characteristics that have been redefined.

Icons, Bitmaps, and Pointers are also implemented as objects. These can be applied to controls and windows as desired. This procedure can be a great time-saver, especially when debugging the code for assigning a bitmap as a menu pull-down. From a programmatical standpoint, these activities are not difficult; the difference is that Gpf makes the task exceedingly easy.

User Function Objects are discussed in the next section.

ENTERING YOUR OWN SOURCE CODE

Gpf provides you with a number of different vehicles through which you can hook your own custom source code into the application definition.

The primary vehicle that Gpf gives you for including your own custom code is User Function Objects. These objects either contain raw source code, or references to code located elsewhere. When you direct Gpf to generate code, this code is inserted into the appropriate place in the program,

which then gets compiled. These objects are also attached in the desired location through Actions. In this case, you specify User Function when you select the action type. There are three different methods for entering your code into the application through User Function Objects:



Figure 5. Defining a Presentation Object

The first method is to actually type the code directly into the multiple line entry field that Gpf gives you. Because it supports the Clipboard, you can paste in code that was created elsewhere.

The second method is to enter a single function call instead of typing in the code directly such as:

```
UserFunctionCall();
```

which implies source code written elsewhere.

The third method is to macro include the code like in the following example:

```
#include "userfunc.c"
```



Tremendous productivity gains will be realized using Gpf

There is a problem, however, with using these last two methods. Because the files you create the code in are not part of the application definition, no dependencies for them will be generated in the make file Gpf builds for your application. Similarly, no object references for them are inserted into your link statement. While these are mentioned as major methodologies for custom code entry in the Gpf User Guide, it is clear that they hadn't been intended as the integral way to include code in your application definition. Maybe in a future release you will be able to specify an external source code file name, which will then be integrated into the make file.

Two other methods exist for including custom written code in your program:

The first is creating a .H header file with the same file name as the one used for code generation in the application definition. Gpf will insert a reference such as:

```
#include "test.h"
```

into its generated code. This is a generally frowned upon programming practice, both in the Gpf and non-Gpf world, but it is there and technically it does what it is supposed to.

The last method is the hardball approach: build as much of your application as you can through Gpf, generate the code once, and never touch the application with Gpf again. This leaves you with the ability to modify all of the Gpf generated code, with the knowledge that if you go back to Gpf, all your work will be wiped out the next time you run the generate program. This most certainly pulls Gpf out of the development cycle, but there may be instances where you want to do this. About the only things that might fall into this category would be the development of program skeletons, or perhaps the use of Gpf to show you how to write the code used to make a certain kind of control.

Personally, I find custom code entry to be one of the weak points of code-generating CASE tools in general, and unfortunately Gpf is no exception. Some tools take the approach of giving you markers in the generated code that show you where you can insert your own code; during later runs

of the code generation program this code is preserved. I find this to be conceptually clearer when you are writing the meat of your application, which you must do eventually.

In Gpf's case, the only integrated way to put your own code in is to type the code directly into the User Function Objects. This works OK in simple cases. Practically speaking, however, when you are programming, you frequently will need to refer to variables and other similar information that can't be viewed from the vantage point of a User Function Object, which makes your task ponderous. You also want to be able to insert code anywhere you want, and to be able to substitute entire "methods," just like the object oriented languages do.

I think that the most we can expect in the short term would be to have markers that can be inserted into the generated source code that would allow you to place your own code anywhere in any of the generated files, and allow it to be safely preserved during later iterations of the code generator. Maybe someday we can see Gpf generating code in C++!

SQL SUPPORT

One of the more surprising features of Gpf is the way that it can seamlessly integrate certain SQL database calls into the application. When the Gpf program is started, one of the first things that it does is to automatically search your hard disk for databases created previously by the IBM Database Manager, included with Extended Edition. It then offers you a choice of database for inclusion in the application through the use of a PM combo box control.

Currently SQL activities may only be achieved in the form of actions, and the actions include:

- Listbox Load List/Combo Box (SQL), which fills a list box with table column headings.
- Listbox Get Selection (SQL), which retrieves the row associated with the selection in the list box.

While limited in scope, they are a terrific start, and they even work in the Animate function (shown in Figure 6), which lets you test out your application definition before you generate and compile.

Eventually this could probably be expanded to include embedded SQL calls as well, which would make it even more terrific.

RETROFITTING RESOURCES INTO GPF

In the event you want to convert an existing Presentation Manager application for use with Gpf, a program is provided that can convert an existing resource (.RES) file into a Gpf application definition. This utility, GPFRES.EXE, will convert the .RES file into a Gpf .ORC file (the file format which holds the application definition), which can then be edited using the Gpf Editor. This program does a fairly good job, with some resultant clipping or size reductions that can easily be fixed with the Gpf Editor.

ANIMATING AND GENERATING CODE

At any point while you are working on an application definition, you may test your work. Gpf provides two different testing tools from the menu bar: Toolkit Test and Toolkit Animator.

Toolkit Test is a quick and dirty tester that lets you try out very rudimentary items such as pull-down menus. It is useful when you want to try out some simple things and don't want to interrupt your work very long.

Toolkit Animator is an industrial-strength testing facility that actually spawns off a run-time program with your application definition, allowing you to run it like a real, compiled application. The only things that can't be tested with the Animator are your custom written code, and the help subsystem. Everything else runs, including the SQL calls. This is one of the truly outstanding features of Gpf, and can save a great deal of testing time by allowing you to test thoroughly without having to do a compile.

The Toolkit Generate pull-down spawns off an application that generates all of the code needed to compile a running, debugged application, including make file. This program runs off the saved (to disk) application definition. Code is generated for the compiler you specified during installation.

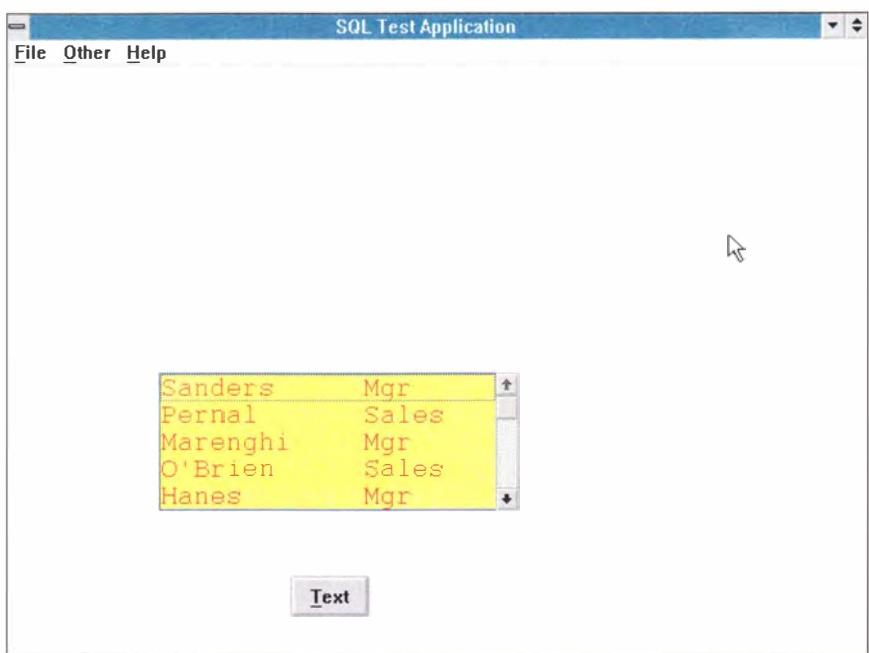


Figure 6. Testing an Application Using Toolkit Animator

OTHER FEATURES

A number of customary features are included allowing you to modify the general appearance and operation of your Gpf session. These include: a snap-to grid for aligning controls, which can be made visible or not; a tracking window that displays size and location coordinates of the selected control; a group move function; suppressing the display of CUA deviation warning messages; and finally, a window displaying statistics and resource usage by the current application.



Gpf can be used as a code generator and as a prototyping tool

USES FOR GPF

This product can be effectively used for two purposes: as a code generator and as a prototyping tool.

As a code generator, Gpf excels in its ability to free the programmer from most of the drudgery associated with the creation of PM windows and controls. It is able to create all of the ancillary files needed to compile an application, even Help and SQL. By combining bug-free code generation with an extremely powerful testing facility, GPF enables most programmers to easily see their productivity enhanced.

Aside from being able to create PM windows and controls, Gpf is also very useful as a prototyping tool. With the ability to hook various windows together and specify Actions to occur as a result of control manipulation such as buttons and pull-downs, applications can be modeled and tested without having to run a compiler or even write a single line of code.

While these are certainly powerful capabilities that can take you a long way down the application development life cycle, effective use of any CASE tool of this type requires a thorough knowledge of straight Presentation Manager programming. As the Gpf User Guide states, "auto mechanics can now hook up a diagnostic computer to a car engine to help them ascertain a problem, but they still must know how to change the plugs themselves."

CONCLUSIONS

While this first cut is already a fine product, there is certainly some room for future improvement. Despite the surprisingly few shortcomings mentioned in this article, it is certainly well worth its price. There is no question that tremendous productivity gains will be realized by any programmer using it. I personally hope that Gpf does well enough in the U.S. market so that we can all watch it successfully mature from this exceptional initial start.

PUBLISHER

Gpf is produced by Microformatic, S.A. The product has recently been announced here in the United States by their U.S. representative, Microformatic U.S.A. Inc., located in rustic Moodus, Connecticut.

The official release of Gpf in the U.S. was on March 1, 1991. Initial response to Gpf by European software houses was so strong that Microformatic was asked to sell beta releases of the software before the manuals or help facilities were available. I suspect that it will do well here also.

The retail price of Gpf is \$3500. Microformatic U.S.A. Inc. can be reached at (203) 873-1950 or faxed at (203) 873-2171. Their address is Box 571 - 26 Plains Road, Moodus, CT 06469.

Michael Drapkin, *Lodestar Software Development Inc.*, 1862 Morningview Drive, Yorktown Heights, NY 10598. (914) 962-1010. Mr. Drapkin is Vice President of a New York City-area software house that specializes in application development using C language under OS/2, DOS, Windows and Presentation Manager. He has performed computer consulting and programming for firms such as American Express, IBM, UPS, Dun & Bradstreet, among others. A renegade musician, Mr. Drapkin received his Bachelor of Music (AMu) from the Eastman School of Music in Rochester, New York.



32-Bit OS/2



OS/2 2.0 Considerations

DISCLAIMER

Much of the information in this article concerns future products, or future releases of products currently commercially available. The discussion regarding Windows is based upon information which the Microsoft Corporation has made publicly available, and is subject to change. The descriptions and discussion of IBM's future products, performance, functions, and availability are based upon IBM's current intent and are also subject to change.

by Claus D. Makowka, Ph.D.

INTRODUCTION & OVERVIEW

Microsoft Windows™ 2.0 has become an important presence on the desktop. You may be wondering, "Why should I use OS/2 2.0 when I can join the Windows phenomenon?" Let's be frank: Windows 3.0 is a significant product and satisfies 5 key desktop needs better than OS/2 1.3:

- Better technology for running the installed base of DOS applications on 386 and 486 processors
- A more polished user interface
- Support for installing as a LAN requester
- Availability of a significant number of Windows applications
- The low cost of upgrading to Windows

OS/2 1.2 and 1.3 provide more subtle advantages:

- Better exploitation of the protection capabilities of Intel™ processors leading to a more robust system
- Smoother operation due to preemptive,

priority dispatching and advanced system facilities available to application developers

- Superior communications capabilities under all combinations of system loads
- Availability of a broad range of powerful database engines.

This leads to the conventional wisdom that Windows is the choice for the desktop while the place for OS/2 is on the server. But technology keeps marching on, and with OS/2 version 2 the balance will change again. Up to now you had to choose between the advantages of Windows and the advantages of 16-bit OS/2. This article shows why that will no longer be true for 386 or better workstations.

First, the objective for OS/2 version 2 is to meet or surpass the Windows advantages.

- Equal capability to run multiple DOS applications while providing a superior environment with better usability, more available memory, and protection against system crashes caused by the application.
- Evolution to a more intuitive user interface for managing the system.
- Support for installing as a LAN requester. Purchase of Extended Edition for the workstation is no longer required.
- Ability to run existing Windows applications out of the box while providing an environment with better system integrity and a more capable file system.
- Competitive price and competitive price to upgrade.

Second, OS/2 version 2 will retain the advantages of earlier OS/2 releases and will add the benefits of exploiting the capabilities of the 386 and 486 processors for maximum performance:



Claus D. Makowka

Why should I use OS/2 2.0 when I can join the Windows phenomenon?



OS/2 2.0's new shell will be a significant step forward

- Dramatic performance improvements are possible for 32-bit applications resulting from use of 32-bit instructions and the large, flat 32-bit memory model.
- More efficient memory use resulting from demand paging.

The purpose of this article is to provide current information about OS/2 and Windows useful to you in making an informed decision. It covers features, market positioning, application migration, and future directions.

FEATURES COMPARISON

At first blush, Windows 3.0 and OS/2 2.0 appear similar. Both have a graphical user interface (GUI), enable applications to take advantage of the larger memory in protected mode, provide concurrent execution of multiple applications, and support execution of multiple DOS sessions. Seeing the differences takes getting to know them a little better. In the interest of making the information as current as possible, the expected capabilities of Windows 3.1, as made public by Microsoft, will be used in this comparison.

USABILITY

Both Windows 3.1 and OS/2 2.0 will be very usable products. Both provide:

- Graphical installation that reaps the benefits of the GUI from the word go.
- An object-based interface with a drag and drop environment that is consistent across the system.
- Support for accessing LAN resources built directly into the shell.
- Intelligent font technology.
- Support for a large number of printers.
- An interactive tutorial.
- Simple productivity applications, games, and utilities included with the system. This lets you learn the system and get to work right away. OS/2 2.0 will include a superset of those provided with Windows 3.1.

However, there are differences. Some usability features unique to OS/2 2.0 are:

- An enhanced user interface, known as the

OS/2 new shell, built on Presentation Manager™ (PM) and around the notion of objects.

- Pervasive online and context sensitive help for the system.
- Support for long file names in the High Performance File System.
- File system support for object-oriented features by means of extended attributes (EAs).

Of all these differences, the most significant for use of the system over the long haul will be the OS/2 new shell. This represents a significant step forward in the GUI desktop environment. The current OS/2 1.3 shell consists of 5 logical units: Desktop Manager, File Manager, Print Manager, Control Panel, and Task List. The shell is being redesigned for OS/2 2.0 to give you a single interface to manage multiple types of objects, including devices (printers and drives), files, and programs. Each defined printer or attached drive is a separate icon. These objects can be arranged at will on the desktop or in specific shell windows. The user will interact with the objects using a well-defined drag and drop environment and will be able to manipulate files without needing to be concerned about the file directory hierarchy. However, if the user wishes, multiple directory trees will be accessible simultaneously. Views, selection techniques, and actions are consistent throughout the shell. The contents of the desktop and shell are saved at shutdown and restored at start-up, preserving the continuity of the work. Finally, applications will be capable of being integrated with the shell. An application object can be the source of a drag to the shell, and an application window can be the target for a drag from the shell. When you act on an application object from the shell, the associated application defines how the object will respond via a dynamic link function that it provides. The shell provides default handling for most actions on file objects.

SOFTWARE TECHNOLOGY

Let's continue by discussing how some of the underlying technologies in Windows 3.1 and OS/2 2.0, as listed in Table 1, affect use of the system. Windows is configured in its 386 enhanced operation mode, which maximizes its capabilities and is the default on a 386 or 486 system with at least 2 megabytes (Mb) of memory.

	WINDOWS 3.1	OS/2 2.0
Physical memory limit	> 16 MB	> 16 MB
Virtual memory limit	4 X Physical	512 MB (disk space)
Memory model	Segmented (64KB)	Flat memory objects
Multitasking - DOS applications	Time slicing	Preemptive time slicing
Multitasking Windows/PM applications	Cooperative	Preemptive time slicing
Priority	Static (set by user)	Dynamic
Dispatchability	Process	Thread
System services	Serial	Parallel
Protection between applications	Unprotected	Protected
Kernel protection — DOS applications	Unprotected	Protected
Kernel Protection — Windows/PM applications	Protected	Protected
File system	FAT	Enhanced FAT High Performance FS Installable
Reliability/availability/ service support	None Included in 3.0 Unknown for 3.1	Standalone dump Error Logging Trace Utilities

Table 1: Windows 3.1 and OS/2 2.0 Technology Comparison

Memory technology is fundamental to understanding how large an application can be, how much data it can handle, and how many applications can be harnessed simultaneously on your PC to get to the solution you need. Answering this simple question for Windows and OS/2 leads into some surprisingly complex technical considerations. The bottom line is that IBM believes that OS/2 2.0 will provide a much larger and more flexible upper limit than Windows 3.1. Let's dig into the details of why this is true.

Systems implementing virtual, paged memory, as both Windows 3.1 and OS/2 2.0 will do, have two types of memory limits that affect how many applications can be loaded and run concurrently: physical memory installed and virtual memory available to running applications. Even though applications only see virtual memory, physical

memory remains important because performance degrades as the amount of virtual memory used begins to exceed the amount of physical memory installed by too large a factor. This occurs because virtual memory is paged into physical memory by the operating system when it is needed, and if there is not enough physical memory the system spends more time shuffling memory pages and less time running the applications. The optimum ratio of virtual memory to physical memory is very dependent on the way the applications are written and used.

While Windows 3.0 supports a maximum 16 Mb of physical memory, both Windows 3.1 and OS/2 2.0 are intended to support greater than 16 Mb, up to the physical limits supported by the hardware. This provides plenty of room for growth since 3 to 6 Mb is expected to be the amount of physical memory needed for the typical use of

32

*The worst
thing a DOS
application can
do on OS/2 2.0
is hang up itself*

Windows 3.1 or OS/2 2.0. For Windows, the maximum amount of virtual memory shared across all applications is limited to four times the amount of physical memory installed, subject to available disk space. In OS/2 2.0, the important limit on virtual memory will be the amount of available disk space.

Technically, each individual application running under OS/2 2.0 will have a 512 Mb limit, with up to 240 applications and with no other overall limit on the total virtual memory used.

Just as important as the maximum virtual memory is the memory model for managing memory. Windows 3.1 apparently will continue with use of a segmented memory model dealing with memory as pieces in any size up to 64 Kb. OS/2 2.0 deals with memory as complete objects of whatever size needed, up to the size of virtual memory. The segment size in Windows constrains the maximum number of many resources used by applications or Windows itself. A simple example is the number of windows that can be simultaneously open within an application, typically about 10. In Windows, these maximums are on an equal footing with the maximum virtual memory in limiting the number and size of applications. The flat 32-bit memory objects provided by OS/2 2.0 avoid the constraints imposed by segmentation. While resource maximums still occur in OS/2 2.0 for other reasons, they are larger and not the result of a fixed constraint imposed by the memory technology.

While the memory technology in Windows 3.1 should continue to provide relief from many of the constraints imposed by DOS, Windows 3.1 will retain memory technology limits that are less than the 386 hardware capability. OS/2 2.0 memory technology takes advantage of the hardware. For example, consider a system with 4 Mb of physical memory. Under Windows 3.1 there will be an absolute limit at 16 Mb of virtual memory. If your applications need even 1 byte more, your applications can't continue. With OS/2 2.0, there will be no absolute limit at 16 Mb, although with only 4 Mb of physical memory you would reach some performance limit before attaining the virtual memory limit of 512 Mb. However, in OS/2 2.0 this performance limit will be soft, in the sense that performance slows down as you need more virtual memory, but you can continue. I have oversimplified a little, inasmuch as limits other than virtual memory size may also apply, but the sense of the discussion remains the same. OS/2 2.0 should

enjoy a significant advantage in removing memory as a constraint on what application developers can achieve.

The remaining technology topics are easier to discuss. The series of items on tasking, priorities, dispatching, and system services are all closely related. The significance is the smoothness with which the system operates and the responsiveness that applications can exhibit. On systems under a light work load, both approaches work well. The OS/2 2.0 technology should hold up better as the work load becomes heavy. This is especially important with high-speed concurrent and background communications. The whole point of multitasking technology is to let you continue on to something new while the system is completing the previous task. A simple, instructive experiment will be to spool a large file to the printer in Windows 3.X, from the file manager or a Windows application, then switch to another application and continue working. Compare this with the equivalent operation on OS/2 2.0 and judge the relative smoothness and responsiveness.

OS/2 2.0 is intended to be a very stable platform to use. Protection features of the processor are utilized to protect applications from each other and to protect the operating system kernel from the applications. One deficiency of OS/2 1.3 was the lack of protection from hanging the system due to DOS application errors. That limitation will be removed on OS/2 2.0. Excepting some unusual cases typically seen only in application program development, the worst thing that an application error will do in OS/2 2.0 is hang up the application itself. Moreover, no application has the ability to meddle with any data in memory that is private to another application. Although the Windows 3.X kernel enjoys some protection from applications, Windows applications execute in the same address space and so are not protected from each other. In particular, Windows 3.X applications may encounter a data integrity risk since an addressing error in one application could modify data private to another application and be undetected. Likewise, key portions of the Windows 3.X kernel can be overwritten by an errant DOS application, resulting in the unknown application error (UAE) message and a request to reboot the system. The importance of having good protection will be clear to anyone who has lost work from multiple applications because one of them managed to hang the system.



OS/2 2.0 will have advantages in its file system design. First, file systems are installable, simplifying support of new storage media requiring specialized functions. An example is the installable file system for

CDROM. Second, OS/2 2.0 will provide the High Performance File System. Major features are advanced strategies for laying files out on disk to minimize fragmentation and maximize disk performance, exploitation of SCSI

	WINDOWS 3.0 ON IBM DOS 5.0					
	IBM DOS 5.0	OS/2 1.3	Real	Standard	Enhanced	OS/2 2.0
Conventional memory with EMS & mouse	623 KB 601 KB	529 KB	558 KB	571 KB	569 KB	633 KB
Memory w/LAN attachment	543 KB	486 KB	478 KB	491 KB	489 KB	633 KB
Extended memory (XMS)	16 MB	none	16 MB (total)	16 MB (total)	16 MB (total)	16 MB (per app)
EMS 4.0 memory	16 MB	none	16 MB (total)	none	16 MB (total)	32 MB (per app)
Physical RAM for DOS	0-1 MB	0-640 KB	0-1 MB	0-1 Mb	Any/Paged	Any/Paged
Memory overcommit	None/ Switch	None/ Swap	None/ Switch	None/ Switch	4 X RAM	Avail Disk
Swap File	File System	File System	File System	File System	Physical or File System	File System
Number of DOS apps	16	1	16	16	16	>32
Background Execution	No	No	No	No	Yes	Yes
Invocation	Shell/Cmd	Icon	Icon	Icon	Icon	Icon
Windowed	No	No	No	No	Yes	Yes
Cut & Paste	No	No	Yes	Yes	Yes	Yes
Print spooling	Yes	Yes	No	No	No	Yes
Installable file system	No	Yes	No	No	No	Yes
Direct H/W access	Yes	Yes	Yes	Yes	Yes	Yes
Timing-dependent applications	Foreground	Foreground	Foreground	Foreground	Exclusive Mode option	Foreground/ Background
DOS Protected-Mode Interface	No	No	No	No	Yes	Yes
Virtual Control Program Interface & DOS Extenders	Yes	No	No	No	No	No
Continues after Serious Application Errors	Rarely	Rarely	Rarely	Rarely	Sometime	Usually

Table 2: Comparison of DOS Environments (Typical values for an IBM model 8580-071)

hardware performance features, sophisticated caching for reading and writing, support of very large disks, and support for long file names. An enhanced FAT file system will also be supported for floppy diskettes and for compatibility with existing hard disks already using FAT. Numerous performance improvements have been made relative to OS/2 1.3 and DOS while maintaining compatibility on the disk or floppies.

Standard operating system support for Reliability/Availability/Service (RAS) will be another attribute of OS/2 2.0. Utilities will be provided to allow software problems in the system to be isolated and reported. The benefit is that software service is provided in support of reliable operation and availability. Windows 3.0 did not include any RAS-like features. Although Microsoft has announced the intent to offer RAS-like features in Windows 3.1, what will be supported in that time frame is unknown at this time.

As an environment OS/2 2.0 should provide many advantages, both in usability and in the robust technology underlying it.

DOS APPLICATION SUPPORT

Due to the large base of available DOS applications, I would like to focus on a detailed comparison of the DOS application support. See Table 2 for an analysis of the key environments, including DOS and OS/2 1.3. I will use Windows 3.0 for the table because I can make several key measurements on it. Several comments are in order. The table applies to operation on a 386 or better. Normally, Windows 3.0 is not run in real or standard mode on this class of machine, especially if the emphasis is on the DOS environment. However, I include those environments for reference purposes because real mode is needed to run Windows applications that have not been upgraded to support Windows 3.0, and standard mode may give better performance in systems with limited memory.

The first group of entries relates to the raw memory space seen by the DOS application. Conventional memory is the memory remaining to the application after subtracting the memory overhead used by the system code. The values given are typical for a default installation. IBM DOS 5.0 appears to have more conventional memory because the default DOS installation does not include

expanded memory (EMS) or mouse support. In Windows 386 enhanced mode, the default installation provides EMS support. The OS/2 2.0 default installation will provide both EMS and mouse support. Including support for EMS (at 8 Kb) and mouse (at 14 Kb) reduces the available conventional memory in DOS to 601 Kb. In addition, any resident software or device drivers for functions such as LAN or host connectivity are subtracted from the conventional memory available to DOS applications under DOS or Windows, but do not affect the memory available in OS/2 2.0. This is illustrated by an example showing memory remaining with a LAN requester installed. EMS support is through emulation of the EMS 4.0 specification out of extended memory. Note that Windows 3.0 has a fixed pool of memory allocated among all applications, while OS/2 2.0 will have a separate, larger, limit for each individual application. The OS/2 2.0 limit on extended memory that will be available to a DOS application is based on using the extended memory specification (XMS) interface. DOS applications using the DOS Protected Mode Interface (DPMI) specification will be able to access up to 512 Mb of extended memory.

The next group of entries covers memory management characteristics. Physical RAM describes the physical memory locations that are available for loading the DOS application. Where only memory below 1 Mb is available, only one DOS application can be in memory at a time, and the other DOS sessions are swapped out to disk. Memory overcommit is the capability to run applications needing more memory than is physically available on the machine. The "none/switch" notation means that no individual DOS application can overcommit memory, but the real mode portion can be moved to disk to make room for another DOS application. However, extended or EMS memory allocated by the application is not switched to disk. OS/2 1.3 can swap the DOS application to disk when running protect mode applications. Windows 3.0 386 enhanced mode can overcommit up to 4 times the physical memory on the machine. OS/2 2.0 will be limited only by the amount of available disk space. Although the default is to swap through the file system, Windows 386 enhanced mode does allow a swap space to be preallocated, which leads to improved performance by avoiding the DOS file system. Because all of this disk space is preallocated, none of it is available to be shared dynamically for any other use. OS/2 2.0 will implement access to the swap space via the file

system for both FAT and the High Performance File System. The OS/2 2.0 implementation should provide the benefit of a dynamically sized swap file combined with good performance.

Continuing down the table, the upper limit on the number of DOS applications that will run under OS/2 2.0 will be 240, equal to the maximum number of applications. In practice, few people will even reach the suggested limit of 32. OS/2 2.0 will provide windowing of DOS applications on the PM desktop in all text and VGA graphics modes while Windows 3.0 386 enhanced mode supports windowing of DOS applications for text and CGA graphics. Although Windows does include a print spooler, its services are not available to DOS applications and printing concurrently from DOS applications is not supported. Windows permits only one DOS application to print and requires that other DOS applications be suspended if they attempt to print concurrently. Use of a DOS print spooler (loaded prior to Windows) is not a viable solution since printing concurrently from multiple DOS sessions will cause all the output to be jumbled together on the same page. Windows does warn of a device conflict in use of the printer in the latter case and offers a choice on how to proceed, but independent of the choice made, the same jumbled output results. OS/2 2.0 provides correct spooling of printer output from concurrent DOS applications.

Under DOS many specialized devices are supported directly from the application without a device driver. The ability to run these applications is under the heading Direct H/W access. Similarly, some DOS applications are extremely timing sensitive, mostly in the area of communications applications using high data rates. These are supported in all environments when the application is in the foreground. However, in DOS, OS/2 1.3, Windows real mode, and Windows standard mode, the application is suspended while in the background and so cannot satisfy any timing constraint. In 386 enhanced mode, Windows 3.0 provides the option of setting exclusive mode, so you can run a timing sensitive application in the foreground without interference due to time slicing. However, all other applications are suspended and do not run while this DOS application is running in exclusive mode. In some cases you can avoid the need for exclusive mode by manually adjusting the relative priorities. The ability to dynamically

manage priorities will permit OS/2 2.0 to multitask timing critical applications both in the foreground and background with minimal attention from you.

Some DOS applications include a technology known as DOS extenders that allow the application to execute out of extended memory. DOS extenders based on the DOS Protected Mode Interface (DPMI) specification are supported under all environments except OS/2 1.3. Other extenders, including those based on the Virtual Control Program Interface (VCPI) specification, are not supported outside of DOS itself. These types of extenders are inconsistent with the mechanisms used in OS/2 2.0 to protect the system or other applications from application errors.

Most of the environments rarely continue after a serious application error because a failure that hangs a DOS application will usually also hang the entire system. In 386 enhanced mode, Windows 3.0 can sometimes continue after a serious application error, but will fail in cases where the application has overwritten portions of DOS or Windows that are in the DOS address space. Moreover, Windows usually advises you to shut down and restart Windows in this situation because of the risk that the errant DOS application may have left the system in an unstable state. Because of the protection mechanisms that will be in OS/2 2.0, a DOS application error should not affect anything in the system other than its own session, allowing OS/2 to continue after application errors.

The bottom line is that when it comes to running DOS applications, OS/2 2.0 should clearly be the superior environment, compared to both Windows 3.0 and DOS.



*OS/2 2.0's
memory
management
is limited only
by disk space*

MARKET POSITIONING

To best understand the marketplace, let's consider the wide variety of workstation environments, usage levels, and needs that exist. Then, let's look at IBM's suggested positioning of the operating systems in relation to those needs.

ENVIRONMENT

Although actual use of personal computers varies widely, it is helpful for these discus-

32

sions to classify personal computer use into four workstation environments as shown in Table 3.

Also, there are various levels of usage for personal computers. We described them with the following usage level characteristics:

- Entry: Uses personal productivity applications one at a time. May have a single asynchronous, LAN, or SNA communications session. System typically has 2 Mb or less of memory.
- Intermediate: Uses personal productivity applications with expanded or extended memory. Uses task switching to switch between multiple applications. May have a single asynchronous, LAN, or SNA communications session. System typically has 2-4 Mb of memory.
- Advanced: Uses multitasking, mission critical or complex applications as well as personal productivity applications. May use one or more simple or advanced communications sessions. System typically has 4-8 Mb of memory.
- Server: Provides broad function, from print/file sharing to distributed/cooperative processing. Additional requirements include interoperability, multitasking, network management, and software distribution.

PRODUCT POSITIONING

Given customers' current confusion regarding the different choices in operating systems, the following guidelines provide a view for your consideration:

DOS: DOS continues to grow as the low-end operating system of choice for entry systems typically equipped with less than 1 Mb of memory. DOS should be selected for entry usage in all four workstation environments.

DOS/WINDOWS: Windows 3.0 provides an excellent extension to DOS for systems containing 2-6 Mb of memory. This graphical Windows environment will continue to play a role in satisfying entry level needs for the foreseeable future. Also, given the significant investment in 8088, 8086, and 286 technology, DOS/Windows provides a means to maximize the investment return in the short term until the move is made to 32-bit technology. DOS/Windows should be selected for entry usage in all four workstation environments and for intermediate usage in small businesses in Standalone Workstation and Work Group LAN environments.

OS/2 1.3 (16-BIT): OS/2 1.3 provides a robust operating system platform for systems containing 2-6 Mb of memory. OS/2 1.3 should establish itself as the high-quality, preferred operating system to run and

<p>Standalone Workstation</p> <ul style="list-style-type: none"> • No LAN attachment • May have asynchronous communications only • Personal productivity and standalone versions of small business, department or special purpose applications <p>Enterprise Workstation</p> <ul style="list-style-type: none"> • No LAN • Controller-based connection to host (mid-range, mainframe) • Local personal productivity and host applications/mail 	<p>Work Group LAN</p> <ul style="list-style-type: none"> • LAN-based resource sharing • Multiple work group LANs (via bridges) • Non-SNA host communications. • Multi-vendor hardware and software • Local and distributed personal productivity, mail, small business, department, and special purpose applications • Server-based systems management <p>Enterprise LAN</p> <ul style="list-style-type: none"> • Large LAN networks • High-volume, high-performance networks • SNA host communications • Multi-vendor hardware and software • Work group applications distributed host applications/mail • Host controlled system and network management
--	---

Table 3. Workstation Environments

develop mission critical applications and large/complex personal productivity applications for users of 286, 386, or better systems. In addition, OS/2 1.3 will continue to be a Systems Application Architecture™ (SAA™) platform and will be made more flexible by packaging the Communications Manager, Database Manager and LAN function (today only available in IBM's OS/2 Extended Edition) so that they will run on any Intel-based hardware platforms using IBM or non-IBM OS/2 Standard Edition. In the future, IBM intends to enhance OS/2's Communications Manager and Database Manager components by adding functions like ISDN, forward recovery, and access to DB2™ (Remote Unit-Of-Work gateway). OS/2 1.3 should be selected for advanced usage in all four workstation environments and intermediate usage in medium/large accounts in all workstation environments. It should also be selected for servers.

OS/2 2.0 (32-BIT): OS/2 version 2 will become the "integrating platform." Multiple DOS, Windows, and 16-bit OS/2 applications will run better on OS/2 version 2 than they run on their current environment for any 32-bit system (386SX, 386DX, 486) containing 3-8 Mb of memory. Communications Manager, Database Manager, and the LAN function will also be available on this platform. Additionally, new 32-bit applications can be developed and run, finally leading to a fuller exploitation of the 32-bit hardware technology. OS/2 2.0 should be selected for intermediate and advanced use in all four workstation environments, and therefore is a candidate for any 386SX or better with at least 3 Mb. It should also be used for servers.

APPLICATION MIGRATION

Given the huge number of DOS applications and the growing numbers of Windows applications, a key consideration is how well this application base runs on OS/2 version 2. This includes both the ability to continue using existing applications and the option of using unique applications that may appear first under DOS or Windows but not OS/2. This requirement needs to be addressed from two viewpoints: user and developer.

USER PERSPECTIVE

The goal for OS/2 version 2 is to run DOS, Windows, and OS/2 applications better than DOS, Windows, or OS/2 1.3, in terms of better usability, better integrity, and equal or better performance. You shouldn't need to upgrade any applications. After installing OS/2 2.0 and spending a short time learning its functions, you should experience an increase in productivity. Migration to OS/2 applications for additional productivity gains can proceed at whatever pace is appropriate to your business needs. IBM expects to make significant progress towards achieving its stated goals in OS/2 2.0. However, these goals may not be fully achieved until a later release of OS/2 version 2. Let's begin by looking at DOS applications, and then continue with Windows and OS/2 applications. Here is how they will operate on OS/2 version 2.

DOS APPLICATIONS: All DOS real mode applications are supported. The key DOS extensions for EMS, XMS, and DPMI services will be provided by OS/2 2.0. Applications using DOS extenders to run in protected mode will work only if the extender conforms to the DPMI specification. When the extender does not conform, an upgrade to a version of the extender supporting DPMI will be required. Most applications will run well without any particular attention from you. In some cases, you will be able to use the advanced properties feature of the DOS environment provided by OS/2 2.0 to reduce resource requirements or improve performance. In a few cases, for applications that are very hardware-, timing-, or DOS-dependent, use of advanced properties will be required to run the application. To assist you in configuring the advanced properties for your applications, we intend to provide a database of recommended settings with OS/2 2.0. The OS/2 new shell will allow DOS applications to be started and managed directly. DOS applications running in all text or VGA graphics modes can appear in windows on the PM desktop. Concurrent printing from multiple DOS applications will be supported by the OS/2 2.0 spooler, unlike Windows 3.0 which does not provide print spooling to DOS applications.

WINDOWS APPLICATIONS: Windows 2.1 and Windows 3.0 applications are supported. All the necessary code will be included with OS/2 version 2, including the standard Windows printer and plotter drivers. For



*OS/2 version 2
will become the
integrating
platform*

other devices you need the Windows printer or plotter drivers provided by the hardware manufacturer. The goal is for Windows 3.0 applications to run together with other applications on the PM Desktop. However, this goal may not be fully achieved until a later release of OS/2. Initially, the Windows 3.0 applications may run on a separate Windows desktop. Windows 2.1 applications will always run on desktops of their own. Windows 3.0 requires shutting down Windows and restarting in real mode to run Windows 2.1 applications. Windows and PM applications will be able to communicate via the DDE and clipboard protocols.

OS/2 APPLICATIONS: Last, but not least, are existing OS/2 applications, which continue to work and should generally perform better on OS/2 2.0 than on OS/2 1.3. Best of all, you are poised to take advantage of new applications as they appear, be they DOS, Windows, 16-bit OS/2, or the new, more powerful 32-bit OS/2 applications.

DEVELOPER PERSPECTIVE

While the decisions for the user are simple and straightforward, the decisions facing the developer are considerably more complex. Should applications be developed for Windows, OS/2, or both? What about future applications? Should I write a 32-bit application? Microsoft has disclosed that a 32-bit version of Windows will be released. What development implications does this have?

DOS APPLICATIONS: Before beginning on the main theme, let's quickly deal with the topic of migrating DOS applications. The bottom line is that the structural design differences imposed by writing to a GUI require considerable rework in the structure of the DOS application, independent of whether the target of the port is a Windows application or a PM application. OS/2 2.0 will offer the option of porting the DOS application to a 16-bit full-screen application with minimal restructuring, just as OS/2 1.3 does. However, given the capability of running DOS applications on OS/2 2.0, the need for such a port is questionable unless the application needs to add function that would benefit directly from access to more memory or from use of features provided by OS/2.

16-BIT APPLICATIONS: Turning to OS/2 and Windows, the reality is that most reliable crystal balls are cloudy at this time. The best

that we can say is there will be a market for both Windows and OS/2 2.0. That is not as painful as it might seem, given the support provided by the Microsoft Software Migration Kit (SMK), sometimes also called the Windows Libraries for OS/2 (WLO). As described in the Microsoft Systems Journal, November, 1990, porting most Windows 3.0 applications to OS/2 is simple. Most applications designed with the port in mind will be able to maintain a single set of readable source code. IBM is developing its own new migration package with enhancements for OS/2 2.0, both to continue to improve performance and to enable missing function, like the palette manager. Performance is key in all of this. Experience with a recent Microsoft SMK is that the overall performance on OS/2 is only about 10 percent slower than on Windows, and performance is noticeably smoother on OS/2 when multitasking. The goal for the IBM migration package is to provide better performance overall on OS/2 2.0 than the same application running on Windows 3.0.

In a related vein, the OS/2 2.0 device driver kit supports porting Windows device drivers to OS/2 as easily as applications port through the IBM migration package. This will make it easy for hardware vendors who support devices on Windows to also support these devices on OS/2. For vendors developing OS/2 - unique drivers, the device driver kit also provides a library of routines and a template OS/2 device driver to simplify the task.

Since OS/2 2.0 will run Windows applications directly, a natural question is, why bother releasing a PM version of the application? In brief, a PM version provides an easy way to differentiate the application by taking advantage of the additional features OS/2 2.0 provides. Examples are:

- Long file names available with the High Performance File System and improved data integrity.
- Better integration into the OS/2 new shell. With moderate effort, participation in the drag and drop environment. With more effort, fully manage objects as an active participant with the shell.
- Use of threads for better multitasking and responsiveness within the application. Gaining the full benefit of this can be complex.



In some cases, the application requirement will be for features best provided on OS/2 2.0. This includes applications with communications and database requirements that are satisfied by using services provided by OS/2 extensions like OS/2 Extended Edition. Other candidates are applications that require the interprocess communication, tasking, semaphore, multithreading, or graphics capabilities provided in OS/2.

32-BIT APPLICATIONS: The most obvious benefit of writing a 32-bit OS/2 application is the flat memory model. In this environment, there will not be all the problems associated with 64 Kb segments. The 16-bit segmented memory model forces applications to write code that depends on segmentation and manages the virtual address space. Applications that have many code and data segments or that require access to very large data objects are especially sensitive to limitations of 16-bit segmentation.

Any discussion of the benefits of a 32-bit architecture leads directly to performance. The 32-bit OS/2 enables exploitation of the 386 and 486. The performance improvement clearly depends on what the application does and how the application interacts with the operating system. Advanced capabilities such as full-motion video, digital sound, speech, and handwriting recognition often require complex algorithms that should be possible to implement much more efficiently on a 32-bit architecture. As demonstrated in preliminary testing of performance sensitive areas of OS/2 2.0, there should be a significant benefit in the use of native 386 instructions with 32-bit operands. Low-level functions such as string manipulation, 32-bit integer math, data movement and pointer operations are approximately twice as fast as 16-bit equivalent instructions. Applications written in C should be able to achieve 5-15 percent improvement with minor changes and recompilation. Simple memory management changes should be able to buy an additional five percent of performance. Applications fully implemented to take advantage of the flat memory and the new unique 32-bit interfaces should be able to achieve 20-60 percent performance improvements. The flat memory allows the elimination of loading and reloading of segment selectors for accessing FAR data, FAR calls, huge arrays, and long variables. The combination of 32-bit applications using the new, optimized 32-bit application programming interfaces (APIs), like memory management and semaphores,

maximizes the performance benefit. An example of an application illustrating the greatest performance improvement seen to date is REXX, where (in preliminary testing) the above changes produced a 60 percent improvement in REXX's performance for processing statements.

A primary motivator behind the design of the OS/2 2.0 APIs is portability. The flat, 32-bit virtual address space is a common feature on many platforms and is key to portability. In order to compete with other high-end workstation operating platforms such as UNIX™, 32-bit OS/2, including its applications, dynamic link libraries, and kernel should be able to be retargeted to other processors, for example, RISC™ processors. Likewise, it is desirable that applications from other platforms port easily to OS/2. The 32-bit flat memory architecture in OS/2 2.0 removes one of the key inhibitors to a successful port.

In support of portability, considerable effort has also gone into cleaning up the APIs to

16-BIT OS/2	32-BIT OS/2
Multitasking APIs	Minor modifications Some redundant APIs deleted
Signal APIs	Combined into exceptionmanagement • Simple to change
Exception-management APIs	Process model becomes thread model Exit processing unchanged
Pipes/queue APIs	Unchanged
Semaphore APIs	New, concise semaphore model • Simple to migrate/change
Timer Services	Unchanged
Memory management APIs	New APIs to manage flat memory objects • Simple for most applications
File System APIs	Unchanged (except names standardized)
Session management APIs	Unchanged
Presentation Manager APIs	Minor modifications Some redundant APIs deleted
Device Driver Interface (DDI)	Unchanged

Table 4. PM Application Migration from 16-bit to 32-bit OS/2



make them extensible, to remove dependencies on specific processor architectures, and to provide a meaningful and consistent naming convention. For those contemplating a port of an OS/2 application from 16-bit to 32-bit, Table 4 provides a summary of what you can expect.

Another interesting case is creating 32-bit versions of Windows applications. The decision here is more complex because Microsoft has disclosed that a 32-bit version of Windows will be released. Let's defer the discussion of how Windows 32 fits in the larger scheme to the next section on Future Directions and focus on the porting issues for the moment. Table 5 presents a view of what it means to port a 16-bit Windows application to 32-bit, either Windows or OS/2, broken down by the API groups. The information on Windows 32 is based on the information made publicly available by Microsoft at the time this was written.

16-BIT WINDOWS	32-BIT WINDOWS	32-BIT OS/2
User APIs (Window Management)	All 16-bit values become 32-bit, including message parameter and handles. Care is needed with window words	Similar changes
	Input model changes Message reordering	No change
GDI APIs (Graphics)	32-bit coordinate system Many graphics apps affected	16-bit coordinate system available (32-bit optional)
	60+ API calls changed Some API calls removed	GDI to GPI change similar
Multitasking APIs	All new APIs	Similar changes
Pipes APIs	DOS calls turned into Windows APIs	Similar changes into OS/2 base APIs
Timer Services	??	Use OS/2-based APIs
Memory Management	Change to flat memory	Similar changes
File System APIs	DOS int 21 functions turned in Windows APIs	Use OS/2-based APIs
Session management	Largely unchanged	Use OS/2 PM APIs

Table 5. Application Migration of 16-bit Windows to 32-bit Windows or to 32-bit OS/2

The immediate appeal of Windows 32 is that it should be easy to move Windows applications to this API. A port to OS/2 is perceived to be more difficult. However, this is not necessarily so. Even for the User and GDI APIs, used for window management and graphics, which some perceive as an inhibitor to migrating from Windows to PM, the changes needed are similar for both Windows 32 and OS/2 2.0. The net is that a lot of the pain of moving a Windows application to exploit the new 32-bit world is dealing with the new function. The rest comes with the territory, changing parameters from 16 to 32-bits and from segmented to flat memory. In many cases, 16-bit OS/2 PM applications can be converted to 32-bit with minimum effort, occasionally with just a recompile. Converting 16-bit Windows applications to 32-bit will require work, independent of whether the target is Windows 32 or OS/2. Within the API categories listed in Table 5, the function provided by the Windows APIs closely matches the OS/2 2.0 functionality. However, at this point the semantics of the individual API calls are different. Another consideration is that OS/2 2.0 will provide support for mixed 16- and 32-bit applications, easing the migration process by allowing it to be spread over time. As of this writing, Microsoft is offering Windows 32 support only for 32-bit applications. Finally, most of the function promised for Windows 32 will be available on OS/2 2.0 this year.

LANGUAGES AND TOOLS

IBM's goal is to make OS/2 version 2 the development platform of choice for both OS/2 and DOS/Windows applications by enabling developers to exploit the power of the newer, high performance hardware platforms. This includes the capability of running multiple compiles for program builds and support for debugging DOS and Windows applications. The high degree of system integrity provided by OS/2 version 2 will be an additional benefit. The bottom line should be better performance and throughput, resulting in higher productivity.

IBM is working to ensure the availability of a comprehensive set of programming tools on OS/2, both from IBM and independent software vendors (ISVs). IBM intends to provide a core set of 32-bit development tools including:



- A 32-bit optimizing and ANSI conforming C compiler
- A configurable programming editor (extensible through REXX)
- A 32-bit PM interface debugger
- A project-oriented workbench with an open architecture

IBM is working with ISVs to provide the comprehensive existing 16-bit OS/2 tools in 32-bit OS/2 versions as well. The IBM tools will be provided with IBM service and support. Note that the existing 16-bit OS/2 tools continue to work on OS/2 version 2 for development of 16-bit applications.

No prerequisite toolkits should be needed for the development of simple applications. The necessary libraries and binding files will be shipped with the operating system. The base system also includes the standard IBM Procedures Language REXX. REXX is an easy-to-use language that should become an ANSI standard. The future direction for REXX will take it into the world of object technology to enhance both its ease of use and power. A toolkit will be provided for development of more complex applications. This will include items like resource editors and compilers, message file creation and binding tools, the IPF compiler, system profilers, and online documentation.

FUTURE DIRECTIONS

Many of you are aware of the future plans for Windows as discussed by Microsoft at Fall COMDEX™ and in subsequent reports in the trade press. In this section we will look at future OS/2 directions in this context, comparing OS/2 with Windows as appropriate. Areas of discussion will be the New Technology (NT) kernel, database, multimedia, software components, LAN solutions, and distributed environments.

WINDOWS 32 AND THE NT KERNEL (OS/2 3.0)

In September 1990, the division of development responsibilities between IBM and Microsoft changed. Microsoft accepted lead responsibility for the development of a new, portable kernel based on newly emerging technologies. This kernel is to be

designed to B level security standards and will be certified at the C2 level. New file system technologies and symmetric multiprocessing are also being explored. The plan is to provide a future release of OS/2 built on top of this kernel. The new news at COMDEX was that Microsoft also intends to support Windows and a 32-bit extension of Windows on this kernel. Our understanding is that while Windows 32 will also be supported on DOS, the more advanced Windows 32 capabilities, such as security, fault tolerance, and symmetric multiprocessing, will only be supported on top of NT. Because OS/2 will also be supported on top of NT, the same capabilities will also be available or can be provided easily to OS/2 applications. Most of the remaining function slated for Windows 32 is available in OS/2 2.0 this year. Microsoft is promoting Windows 32 as an easier migration path for porting existing Windows applications to 32-bit implementations. As discussed in 32-Bit Applications earlier in this article, the ease of migration is open to debate.

One of the objectives of NT is to run on non-Intel processors, notably RISC processors. While DOS, Windows (16 and 32-bit) and OS/2 (16 and 32-bit) applications can be easily supported by NT running on an Intel processor, the situation is more complex on RISC. The technology for providing DOS emulation on RISC is well understood and is available on the IBM RISC System/6000*. Windows and OS/2 applications that are 32-bit need to be recompiled, with assembler routines rewritten. For 16-bit applications, whether Windows or OS/2, the memory segmentation model presents a problem. The only fair statement is that the technology for getting these applications to run well on a RISC processor, short of porting them to a 32-bit implementation, is not understood today. For this reason, 32-bit implementations appear to be a better choice today if future portability across processors is a business consideration.

DATABASE

IBM is developing the OS/2 database to meet the industry's need for scalable high performance, robustness, reliability, fault tolerance, manageability, security, recoverability, mainframe database access, and interoperability across multiple platforms. IBM intends to provide full 32-bit operation on OS/2 version 2 and support for client

IBM's goal is to make OS/2 2.0 the development platform of choice



operations under DOS, Windows, OS/2, AIX®, and UNIX. Both ANSI SQL® and SAA SQL compliance are goals, with support for application development in multiple high-level languages on OS/2, AIX, and UNIX. APIs will be developed for use by applications and front-end tools to monitor and control database operations. Work is proceeding towards support of a distributed environment, including links to DB2 and SQL/DS®. The published DRDA protocol is intended to give both IBM and non-IBM developers the ability to provide software that interfaces as either a server or a requester to DB2 initially, with subsequent support on the other IBM relational database products.

MULTIMEDIA

Multimedia will be an important new area on both Windows and OS/2. Development of multimedia products can be easier on OS/2 than on Windows. The experience with IBM products like Audio Visual Connection® and M-Motion has been that Windows implementation requires significantly more effort than OS/2. This is a result of:

- Better encapsulation of system resources by OS/2 1.3. Under Windows the programmer needs to be aware of directly controlling the system resources.
- Better stability of OS/2 1.3, especially in the area of tasking. This provides improved coexistence of applications with less coding effort. For example, yield points required under Windows are not necessary with OS/2 1.3.
- Better graphics, such as Bezier functions, on OS/2 1.3. On the other hand, Windows provides image functions that are planned for, but not yet available on, OS/2. Examples are transparency color maps and bit blit of compressed bit maps.

The combination of better tasking and the support for threads leads to better synchronization of the multiple data streams needed for multimedia. An example is synchronizing digital audio with analog video for a firing cannon or a talking head. OS/2 2.0 fast threads are an enhancement specifically added to improve the performance of threads

for multimedia applications. In Windows, synchronization is left entirely to the application.

Multimedia applications frequently need to deal with large memory objects for bit maps, audio streams, or even streams of bit maps. These objects are much easier to manipulate in the flat 32-bit memory model provided by OS/2 2.0. Although the Microsoft Windows 3.0 Software Developer's Kit (SDK) provides the WINMEM32 DLL, which is an interface for allocating 32-bit flat memory from a Windows application, Windows itself remains a 16-bit segmented environment. Use of WINMEM32 leads to a mixed 16/32-bit application, with the usual risks associated with the unprotected, global memory allocation used in Windows 3.0. As described in the SDK, all the complex issues of managing a mixed 16/32-bit application beyond memory allocation basics must be addressed by assembly language code provided by the developer. Moreover, the WINMEM32 DLL is not part of the retail Windows 3.0 product, nor do we believe that the API will be compatible with Windows 32.

A common standard for multimedia data and the control of multimedia devices was released by Microsoft and IBM on November 26, 1990 as an advanced guideline for multimedia developers.

OBJECT-ORIENTED DEVELOPMENT

A significant, emerging software technology is the use of objects. IBM intends to enhance OS/2 version 2 with extensions to improve support for object-oriented programming languages. Support of class libraries using an approach equivalent to dynamic link libraries (DLLs) is contemplated. Presentation services will be enhanced with more object oriented features. In addition, there will be further object oriented extensions to the user interface.

PATRIOT PARTNERS

In looking ahead at the coming decade, a key challenge in growing the software business is finding a way to profitably provide software solutions for specialized problems rather than the types of general solutions addressed by today's spreadsheet, graphics, word processing and database products. Patriot Partners, a partnership between IBM and Metaphor™ Computer Systems Corporation, was formed with the objective of creating a multiplatform application development environment suitable for addressing this challenge. The partnership product has as key objectives:

- To vastly reduce the development risk associated with selecting a specific hardware and operating system platform by enabling a single product to target multiple platforms from multiple vendors.
- To enable the introduction of new software technologies like object-oriented programming, visual programming, multimedia, and expert systems.
- To provide platform independent objects needed to build applications in an extensible development environment.
- To encourage the development of applications built out of reusable components connected by well-defined protocols. This provides the opportunity to sell the components as well as the application.
- To provide tools known as assemblers and builders for combining components into specialized applications/solutions. The development expense is limited by the ability to purchase and reuse existing components, which should make more specialized applications more affordable.
- To enable better application combination by the end user to solve specific business problems via visual programming and the structure provided by components and protocols.

IBM intends to make the partnership technology available on OS/2 and AIX. Metaphor will provide the technology on other platforms, including other UNIX systems and the Macintosh®.

LAN SOLUTIONS

IBM will continue to strengthen its position as a provider of LAN solutions through enhancements to current IBM products and through partnerships and cross-licensing agreements like the one recently announced with Novell® Incorporated. Potential LAN enhancements are being considered in the areas of local and wide area networking, interoperability, management, and communications support from both IBM and Novell products. Examples are software distribution, selective backup/archive, transporting Netware® IPX/SPX packets across SNA networks, Netware client access to Communications Manager, and supporting Communications Manager and Database Manager access to Netware servers.

DISTRIBUTED ENVIRONMENTS

IBM recognizes the increasing importance of heterogeneous vendor support on LANs and networks, and is committed to helping its customers run their businesses with these systems. In May 1990 the Open Software Foundation™ (OSF™) announced the Distributed Computing Environment (DCE), a selection of technologies aimed at simplifying the work for users and application developers in these complex computing environments. In endorsing DCE, IBM subsequently announced not only support for DCE on AIX but also the intent to extend the Systems Application Architecture (SAA) to incorporate key elements of DCE. As one of the SAA systems, OS/2 will be a platform for the delivery of the key elements of DCE. For those of you not familiar with DCE, it consists of eight key technologies. Of these, six are of particular interest for distributed environments including OS/2, and can be summarized as follows:

REMOTE PROCEDURE CALL (RPC) - The basic notion is that procedures called by an application may actually be run on a computer somewhere else in the network. The RPC mechanism takes care of the communications details so that writing distributed applications approaches the simplicity of applications on a single machine.



*Patriot Partners
was formed
to create a
multi-platform
development
environment*



DISTRIBUTED NAMING SERVICE - This provides a single naming model throughout the distributed environment. Resources such as servers, files, disks, or print queues are identified by name independent of the physical location in the network. Full X.500 support is provided.

TIME SERVICE - Many distributed applications need a single time reference to properly determine event sequencing and duration. The time service provides a mechanism for synchronizing each computer in the network to a recognized time standard.

SECURITY SERVICE - Provides the network with authentication, authorization, and user account management. Authentication validates the identity of a user or service to prevent fraudulent requests. Authorization is the process of determining whether an authenticated user should have access to a resource. These facilities are made available through a secure communications capability provided by the RPC.

THREADS SERVICE - A facility to support concurrent programming much like threads in OS/2. This is used by other DCE components to implement their services.

DISTRIBUTED FILE SYSTEM - Joins the file systems of the nodes in the network through a consistent interface that makes global file access as easy as local file access. The Distributed File System should provide users with a uniform name space, file location transparency, and high availability.

CONCLUSIONS

OS/2 2.0 will provide the features that our customers need. OS/2 2.0 will provide compatibility for existing DOS, Windows, and OS/2 applications along with the performance and future opportunities provided by 32-bit applications. OS/2 2.0 will provide a stable base for the increasingly complex, mission critical, and networked applications that we use. With the OS/2 new shell, OS/2 2.0 will provide an object-oriented user interface to help us be more productive.

Of course, DOS, Windows, and OS/2 all have a place on the workstation. In this mixed environment IBM will provide a simple migration path to preserve investment in existing applications as requirements on the computation platform become more complex. Windows 3.0 provides a superset of the DOS capability retaining the ability to support DOS applications while adding the benefits of the graphical user interface (GUI) provided by Windows applications. With OS/2 2.0, IBM intends to provide a superset of the Windows capability and support for DOS and Windows applications while adding the benefits of an advanced GUI (OS/2 new shell), greater integrity for mission critical applications, support for complex communications requirements, and support for both 16- and 32-bit OS/2 applications. Among these three PC operating systems, OS/2 2.0 with 32-bit applications is the only one that in 1991 will enable the user to utilize the performance benefits of the 386 and 486 hardware platforms. Because of these additional benefits combined with the competitive price, OS/2 2.0 should be explored for any suitable 32-bit hardware configuration in the near term.

SPECIAL NOTICES

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates.

IBM may have patents or pending patent applications covering subject matter in this article. The furnishing of this article does not imply giving license to these patents.

CONVENTIONS

Unless specifically qualified as IBM DOS, any use of DOS in this article refers generically to both IBM DOS and MS-DOS.

This article uses Windows 3.0 and Windows 3.1 to refer to specific releases of Windows, and uses "Windows 3.X" to describe both the 3.0 and 3.1 releases. Windows 32 refers to the proposed 32-bit version of Windows that has been disclosed by Microsoft.

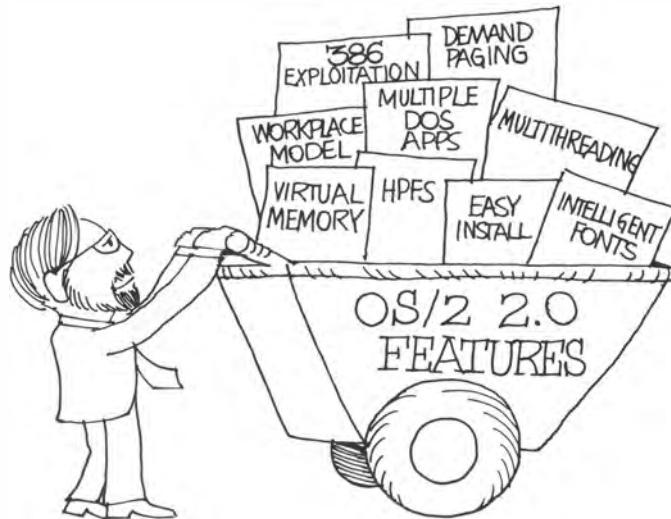
The first release of IBM OS/2 Standard Edition 2.0, is referred to as "OS/2 2.0." The phrase "OS/2 version 2" includes subsequent releases as well.

ACKNOWLEDGMENTS

In preparing this article I have benefited from the work of many thoughtful people throughout IBM who have wrestled with the questions addressed herein. I would like to acknowledge a special debt to the following people, from whose work I have borrowed liberally: Lori Brown, Jack Boyce, Kip Harris, Dick Kamerer, David Kerr, Jim Martin, Debbie Smartt, Mark Tempelmeyer, and Jay Tunkel.

Any errors in the material as presented here are, of course, my own.

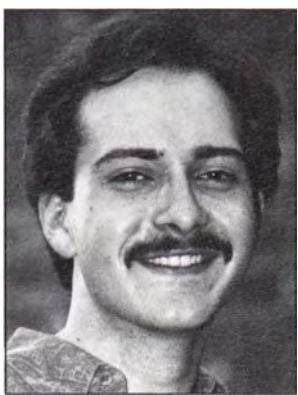
Claus D. Makowka, IBM Personal Systems
Boca Programming Center, PO Box 1328, Boca Raton, FL 3342. Dr. Claus Makowka is a Senior Engineer in Systems Evaluation and Planning. He joined IBM in 1982 and has had assignments developing printer technologies and developing communications subsystems prior to joining the OS/2 team in 1988. He has been an OS/2 designer and manager of OS/2 performance design. Currently, he is responsible for technical evaluation of operating systems and hardware support. Dr. Makowka holds a BS in physics from the California Institute of Technology and both an MS and a PhD. in physics from the University of Illinois at Urbana-Champaign.





Presentation Manager

System Input Hooks under OS/2



Kenneth Tabor

A hook examines message passing through an application queue

by Kenneth Tabor

The "Corporate Help Facility" is a program I developed as a test of the system input hook capabilities of the OS/2 Presentation Manager. Using "The Corporate Help Facility," additional help may be assigned to any window on the OS/2 desktop. This additional help is in the form of a Help Manager panel activated by pressing <Ctrl H>. The help panel may cover any subject and include any information necessary to assist a person in daily computer use. Additional help can be context sensitive since it is linked to a window's title, not just an overall application. By taking advantage of a few advanced programming practices including dynamic link libraries, window processing, Help Manager panels, and system input hooks, it is possible to create a "Personal Help Facility" of your own.

A hook is a function that examines messages passing through a specific application message queue, or PM's system message queue. This is accomplished by a call from PM, or directly by the application, to the hook in response to an event or message. To maintain run-time performance it is important to keep the hook's responses short, since there is a large volume of messages sent to an application during its execution.

An input hook is constructed to examine messages in a queue when they are removed, but before they are returned to the application window. Organized this way, an input hook can be used as an overlay to a window, or a set of windows. Monitoring an entire application is possible with this method.

A good example for this task is in implementing the standard which asks for F10 to activate the main window's action bar. An input hook will see the WM_CHAR

message passing through the application's message queue no matter what window it was posted to. The hook would then post a message to the main client window informing it of the activate menu request. The main window would in turn take the necessary steps to activate itself and set the focus to the action bar.

Another category of hooks is the system wide hook. This type of hook is installed in the system message queue instead of any specific application queue. Hooking into the system queue, by means of a system hook, allows an application the following possibilities:

1. It can act on a message it would not normally receive
2. It can notify one of its windows to perform an action when that window is not active or is not the focus
3. To react to a message in behalf of another window
4. To prevent an application from receiving messages

It is important to note that a system hook must be accessible by all processes. Therefore the hook must reside in a DLL. If you install a hook with the intentions of hooking only your application's message queue then the hook function can be in your executable file as any other function would.

SUPPLEMENTING APPLICATION HELP

In the ongoing effort of end user support it would be advantageous to supplement the help of an application without modifying it. That help should be available through any

window on the desktop, and should be tailored to the user's needs. Support of this personalized help would be the job of the technical support department of the user's company. They will have the best idea of what extra help their people need to know according to business needs. To this end "The Corporate Help Library" gives help about the current window at the press of the <Ctrl H> key combination.

Help can have a variety of meanings. It can be time saving macros, well composed style sheets, miscellaneous tips and tricks. Help can be as easy as pointing a person to the correct page in a manual, or telling them why they should hit <Alt P> instead of <Alt F>. Best yet a list of contact names and telephone numbers for further help can be recorded. All of these valuable points can be written down, collated, and assembled into a number of on-line help panels representing a Personal Help Library.

After installing a system hook, the Help Library application can spend its time running quietly on the desktop, while an innocuous icon waits for the occasion of <Ctrl H>. Once a <Ctrl H> call is made, the hook notifies its application window and returns to its watch. The active window is queried and a search for help on it is performed after its title is found. If the window title is found the corporate sponsored help is displayed by sending the PM Help Manager a request to show the panel.

IMPLEMENTING SUPPLEMENTAL APPLICATION HELP

Figure 1 shows the input hook function. Every message will be examined by this function so speed is a major consideration in its design. The first statement compares the incoming message with WM_CHAR, if a keystroke is hit then it checks the state of the Ctrl key. If it is down then the character is queried for the 'H' key. If it is also found then the user is requesting help and the hook notifies the application window to take over processing.

To find the application window the hook searches all the top level windows on the desktop using window enumeration functions. When it finds a frame window with the window text, "XYZ Corporate Help Library", the hook posts a notification message to the client window. Processing then falls out of the hook with a TRUE.

The TRUE tells OS/2 not to pass this message on to the next hook, or the target message queue. The hook assumes the <Ctrl H> was targeted specifically for it and stops the message after processing. Since <Ctrl H> is the only message the hook wants to see, all others are passed to the next hook or message queue, by returning a FALSE to OS/2.



```
***** Our System Input Hook Source Code *****
/*
 * PROGRAM NAME: CORPHOOK
 */
/* OS/2 DLL portion of the help hook application
 */
/* COPYRIGHT:
 */
/* (C) Copyright International Business Machines Corporation 1990, 1991
 */
/* WHAT THIS PROGRAM DOES:
 */
/* This code module is a dynamic link library installed by the main
 * application as a system input hook. This hook waits for the user to
 * press the <Ctrl H> key combination and notifies the main application
 * window of the event.
 */
/* WHAT THIS PROGRAM DEMONSTRATES:
 */
/* This program demonstrates how to write a system input hook in the proper
 * DLL form. Also, how to find a window handle based on it's window text.
 */
/* REQUIRED LIBRARIES:
 */
/* OS2.LIB      - Presentation Manager/OS2 library
 */


```

Figure 1. Example of the Input Hook Function (Continued)



```

/*
 * LIBCE.LIB      - Protect mode/standard combined large model C library
 */
/*
 * REQUIRED PROGRAMS:
 */
/*
 * IBM C Compiler
 * Information Presentation Facility Compiler (IPFC)
 * IBM Linker
 * Resource Compiler
 */
/*
 * OS/2 APIs to know:
 */
/*
 *      WinBeginEnumWindows      WinWindowFromID
 *      WinGetNextWindow        WinPostMsg
 *      WinQueryWindowText      WinEndEnumWindows
 */
//*********************************************************************//

#define INCL_WINFRAMEMGR
#define INCL_WINHOOKS
#define INCL_WININPUT
#define INCL_WINWINDOWMGR
#include <string.h>

extern _arctused =0;

BOOL EXPENTRY CorpHelpHook( HAB    hab, PQMSG   pQmsg, USHORT   fs)
{HENUM      hEnum;
 HWND       hC, hF;
 USHORT     ch, flags;
 CHAR      str50;

    if( pQmsg->msg == WM_CHAR){ /* Is the event a keypress? */
        flags =SHORT1FROMMP( pQmsg->mp1);
        if( flags&KC_CTRL){ /* Was the <Ctrl> key down? */
            ch =SHORT1FROMMP( pQmsg->mp2);
            if( ch=='H' || ch=='h'){ /* Was the 'h' key also pressed? */
                hEnum =WinBeginEnumWindows( HWND_DESKTOP);
                do{
                    /* Search through all the top-level, desktop, windows
                     ** for the main application window. The frame windows
                     ** will be returned by the enumeration calls, and the windows'
                     ** text will be the title-bar title. A string compare
                     ** is used for verification of the proper window.
                     */
                    hF =WinGetNextWindow( hEnum);
                    WinQueryWindowText( hF, 50, str);
                    if( !strcmp( str, "XYZ Corporate Help Library")){
                        /* When the application window is found notify it
                         ** of the user's request for help.
                         */
                        hC =WinWindowFromID( hF, FID_CLIENT);
                        WinPostMsg( hC, LOOKUP_WM, NULL, NULL);
                    }
                } while( hF);
                WinEndEnumWindows( hEnum);
                return( TRUE); /* Do not pass this keypress event on */
            }
        }
    }
    return( FALSE); /* Do pass this event message on */
} /* end of CORPHELPHOOK */

```

Figure 1. Example of the Input Hook Function

Figure 2 contains the main() code. As with all PM programs it has the standard initialization, window creation, and message looping. Of more importance here are the DLL loading and hook functions.

DosLoadModule() is used to load the DLL with the input hook into memory. WinSetHook() is used to install the hook from the DLL.

If you want the hook to monitor your application's queue, then the second parameter in WinSetHook() will be your HMQ instead of NULL. Also, since the hook is local it would probably reside in your .EXE so no DLL would be loaded, and therefore the last parameter, HMODULE, would be NULL as well.



The application described here is a good example of the power and flexibility of PM

```
***** Our Application's main() Function *****/
/*
/* PROGRAM NAME: CORPHOOK
/* _____
/* Main application function.
/*
/* COPYRIGHT:
/* _____
/* (C) Copyright International Business Machines Corporation 1990, 1991
/*
/* WHAT THIS PROGRAM DOES:
/*
/* This code module is the Help Facility main program. It has the standard
/* OS/2 support code and the main window procedure. This window starts
/* as an icon since it never needs user interaction except for closing it.
/*
/* When a help event is received from the hook the current window is
/* queried and its supplemental help is displayed. If the current window
/* is not registered to this program the table of contents is listed
/* instead.
/*
/* WHAT THIS PROGRAM DEMONSTRATES:
/*
/* This program demonstrates how to install a system input hook from a
/* dynamic link library.
/*
/* REQUIRED LIBRARIES:
/*
/* OS2.LIB      - Presentation Manager/OS2 library
/* LLIBCE.LIB   - Protect mode/standard combined large model C library
/*
/* REQUIRED PROGRAMS:
/*
/* IBM C Compiler
/* Information Presentation Facility Compiler (IPFC)
/* IBM Linker
/* Resource Compiler
/*
/* OS/2 APIs to know:
/*
/* DosLoadModule    WinSetHook
/* DosFreeModule   WinReleaseHook
/*
*****
```

Figure 2. Example of the Main() Code (Continued)



```

#define INCL_DOSMODULEMGR
#define INCL_WINFRAMEMGR
#define INCL_WINHOOKS
#define INCL_WINDOWMGR

VOID main( int argc, char argv)
{HAB hab;
 HMQ hmq;
 QMSG qmsg;
 FRAMECDATA fcdata;
 HWND hwndFrame, hwndClient;

HMODULE ModHand;
CHAR LoadError100;

/* Standard OS/2 PM entry */
hab =WinInitialize( 0);
hmq =WinCreateMsgQueue( hab, 0);
WinRegisterClass( hab, "WCMain", wpMain, CS_SIZEREDRAW, 0);

fcdata.cb =sizeof( FRAMECDATA);
fcdata.flCreateFlags =FCF_TASKLIST | FCF_SIZEBORDER | FCF_TITLEBAR |
FCF_ICON | FCF_SYSMENU | FCF_MINMAX;
fcdata.hmodResources =NULL;
fcdata.idResources =RESOURCE_NOTE;
/* Create the frame window */
hwndFrame =WinCreateWindow( HWND_DESKTOP, WC_FRAME, NULL,
WS_VISIBLE,
0, 0, 0, 0, NULL, HWND_TOP,
0, &fcdata, NULL);
/* Create the main client window */
hwndClient =WinCreateWindow( hwndFrame, "WCMain", (PSZ)NULL,
WS_VISIBLE,
0, 0, 0, 0, NULL,
HWND_BOTTOM, FID_CLIENT,
NULL, NULL);
jj
WinSetWindowText( hwndFrame, "XYZ Corporate Help Library");
/* Start the application as an icon by minimizing it */
WinSetWindowPos( hwndFrame, HWND_TOP, 100, 100, 400, 400,
SWP_SIZE | SWP_MOVE | SWP_SHOW |
SWP_MINIMIZE |
SWP_ZORDER);

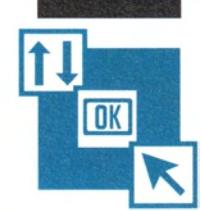
/* Load the DLL with the system hook */
DosLoadModule( LoadError, 100, "CORPHOOK", &ModHand);
/* Tell OS/2 to install the hook as a system input hook */
WinSetHook( hab, NULL, HK_INPUT, CorpHelpHook, ModHand);

/* Standard OS/2 PM processing */
while( WinGetMsg( hab, &qmsg, NULL, 0, 0))
    WinDispatchMsg( hab, &qmsg);

/* Tell OS/2 to remove the system input hook */
WinReleaseHook( hab, NULL, HK_INPUT, CorpHelpHook, ModHand);
/* Free the DLL resources */
DosFreeModule( ModHand);
/* Standard OS/2 PM exit */
WinDestroyWindow( hwndFrame);
WinDestroyMsgQueue( hmq);
WinTerminate( hab);
/* end of MAIN */
}

```

Figure 2. Example of the Main() Cold (Continued)



```
***** Our Application's Main Window Procedure*****
/*
/* PROGRAM NAME: CORPHOOK
/*
/* Main window procedure.
/*
/* COPYRIGHT:
/*
/* (C) Copyright International Business Machines Corporation 1990, 1991
/*
/* WHAT THIS PROGRAM DOES:
/*
/* When a help message is received from the hook the current window is
/* queried and its supplemental help is displayed. If the current window
/* is not registered to this program the table of contents is listed
/* instead.
/*
/* WHAT THIS PROGRAM DEMONSTRATES:
/*
/* This program demonstrates how to use the Help Manager.
/*
/* REQUIRED LIBRARIES:
/*
/* OS2.LIB      - Presentation Manager/OS2 library
/* LLIBCE.LIB    - Protect mode/standard combined large model C library
/*
/* REQUIRED PROGRAMS:
/*
/* IBM C Compiler
/* Information Presentation Facility Compiler (IPFC)
/* IBM Linker
/* Resource Compiler
/*
/* OS/2 APIs to know:
/*
/* WinCreateHelpInstance
/* WinDestroyHelpInstance
/*
*****
```

Figure 2. Example of the Main() Code

When the application is terminated by the user, WinReleaseHook() is called to clear out the input hook. The DLL resources are freed using DosFreeModule().

Figure 3 shows the application's main window procedure. This client window does little more than process requests to the Help Manager.

On WM_CREATE the window initializes the Help Manager and receives a window handle for later communications. On WM_CLOSE the Help Manager session is disconnected. The window spends the bulk of its time waiting for the hook to send it a help notification message.

When the window receives a LOOKUP_WWM, a simple #define of WM_USER+2000, the current window and

its title bar text are queried. Searching through its internal data structure the window procedure attempts to find a help panel id for the active window. If one is found a HM_DISPLAY message is sent to the Help Manager. If the title has no associated help panel then the table of contents is shown instead.

An ability to read an external definition file during the WM_CREATE message processing would be required for locations with multiple installations. The file should contain the search window titles and their respective help panel IDs. A variable window definition and several .HLP files will allow the same executable file to drive help dedicated to several functions. Help files aimed at job description, or departmental function will be easy to install and profitable to use.



```

#define INCL_WINHELP
#define INCL_WINWINDOWMGR
#include "corphelp.h"
#include <string.h>

HWND      hwndHelp;
HELPNODE  table = { "Desktop Manager",           HELP_DESKTOP_MANAGER,
                    "Group - Main",            HELP_GROUP_MAIN,
                    "Group - Utilities",       HELP_GROUP_UTILITIES,
                    "Query Manager",          HELP_QUERY_MANAGER,
                    "OS2 Logo",                HELP_OS2_LOGO,
                    "Control Panel",          HELP_CONTROL_PANEL,
                    "File Manager",            HELP_FILE_MANAGER,
                    "Logon",                  HELP_LOGON,
                    "A - A - 3270 Emulator",   HELP_3270};

MRESULT EXPENTRY wpMain( HWND  hwnd, USHORT  msg,
                        MPARAM mp1, MPARAM mp2)
{
    HWND      hF;
    HPS      hps;
    RECTL   rcl;
    USHORT   id;
    SHORT   rc;
    CHAR    str[81];

    switch( msg){
        case LOOKUP_WWM:
            /* The user pressed <Ctrl>+H and the help hook
             ** has notified the main window with this message.*/
            /* Ask OS/2 what the current window is */
            hF = WinQueryActiveWindow( HWND_DESKTOP, FALSE);
            /* What is its window text */
            WinQueryWindowText( hF, 81, str);
            for( rc = 0; rc < MAX_HELP_ENTRIES; rc++){
                if( !strcmp( str, tablerc.title)){
                    /* The window text in the registration
                     ** table
                     */
                    id = tablerc.id;
                    /* Tell the Help Manager instance to show
                     ** the supplemental help panel.
                     */
                    WinSendMsg( hwndHelp, HM_DISPLAY_HELP,
                                MPFROMP( &id),
                                MPFROMSHORT(
                                    HM_RESOURCEID));
                }
            }
            /* Otherwise the window is not registered, so
             ** default with the supplemental help table of
             ** contents.
             */
            WinSendMsg( NULL);hwndHelp, HM_HELP_CONTENTS, NULL,
                        NULL);
            return( 0);
        }
        /* A simple rectangle fill */
    case WM_PAINT:

```

Figure 3. Is an Example of the Application's Main Window Procedure (Continued)



```

        hps = WinBeginPaint( hwnd, NULL, &rcl);
        WinFillRect( hps, &rcl, CLR_BLUE);
        WinEndPaint(hps);
        return( 0);

    case WM_CREATE:
        /* The application was created, create a Help
        **Manager instance.
        */
        setup_help( hwnd);
        return( 0);

    case WM_CLOSE:
        /* The application was closed, free up the Help
        **Manager
        */
        WinDestroyHelpInstance( hwndHelp);
        WinPostMsg( hwnd, WM_QUIT, NULL, NULL);
        return( 0);

    } /* end of switch...msg */
return( WinDefWindowProc( hwnd, msg, mp1, mp2));
} /* end of WPMAIN */

VOID  setup_help( HWND  hP)
{HELPINIT      hmiHelpData;
 HAB          hab;

 hmiHelpData.cb           =sizeof( HELPINIT);
 hmiHelpData.ulReturnCode =NULL;
 hmiHelpData.pszTutorialName =NULL;
 hmiHelpData.phtHelpTable =NULL;
 hmiHelpData.hmodAccelActionBarModule =NULL;
 hmiHelpData.idAccelTable =NULL;
 hmiHelpData.idActionBar =NULL;
 hmiHelpData.pszHelpWindowTitle = "Welcome to XYZ Company"
                                "Help";
 hmiHelpData.hmodHelpTableModule =NULL;
 hmiHelpData.usShowPanelId =NULL;
 hmiHelpData.pszHelpLibraryName = "corphelp.hlp";
 hab =WinQueryAnchorBlock( hP);
 hwndHelp =WinCreateHelpInstance( hab, &hmiHelpData);

if( !hwndHelp){
    WinMessageBox( HWND_DESKTOP, HWND_DESKTOP,
                  (PSZ) "Help Not Available",
                  (PSZ) "Help Creation Error",
                  1,
                  MB_OK | MB_APPLMODAL | MB_MOVEABLE);
}
else{
    if( hmiHelpData.ulReturnCode){
        WinMessageBox( HWND_DESKTOP, HWND_DESKTOP,
                      (PSZ) "Help Terminated Due to Error"
                      (PSZ) "Help Creation Error",1,
                      MB_OK | MB_APPLMODAL | MB_MOVEABLE);
        WinDestroyHelpInstance( hwndHelp);
    }
}
/* end of SETUP_HELP */
}

```

Figure 3. Is an Example of the Application's Main Window Procedure



```

.* CORPHELP.IPF for CORPHELP.EXE
.*
:userdoc.
:title.XYZ Corporate Help for OS/2 PM
:body.
*
.* HELP_DESKTOP_MANAGER
:h1 res=2000. "Desktop Manager"
:p.This is the main launch pad for OS/2 Presentation Manager. This window
contains a list of this PC's application groups. Each group will contain
several applications to choose from.
:p.In an attempt to organize your application, choices in each group will
list an orderly set of utilities and computer programs. These range from
Group - :hpt.Main:ehpt. :hdref res=2002. to
Group - :hpt.Utilities:ehpt. :hdref res=2003.. Depending on your
application needs you will look into a particular group.
:p.To bring up a group double click on its icon in the Desktop Manager.
Desktop Manager will quickly display the group's program list with the
programs' respective icons. To launch, or run, a program from a group
you must double click on its name or its icon. The application will be
started in its own session.
:p.For more information on the concepts of the "Desktop Manager" and
"Groups" read through the OS/2 EE manual "Getting Started", specifically
chapter four.
.*
.*
.* HELP_GROUP_MAIN
:h1 res=2002. "Group - Main"
:p.The main group will have all OS/2's main applications. Good application
choices to run are the :hp2.OS/2 Command Reference:ehp2. and :hp2.OS/2
System Editor:ehp2..
:p.The Command Reference is an excellent documentation program listing all
the commands in OS/2 EE. If you should have a question pertaining to OS/2
commands I would point to there. If your problems persist try the OS/2 EE
User's Guide Volume 1. If that still does not help then try calling Alex
on x4590.
:p.The System Editor is a nice little text editor meant for quick notes in
the post-it style. Since it will not do much in the way of publishing or
formatting I like to keep it as a clipboard - viewer. If you need help
using the editor call me at x3978.
.*
.* HELP_GROUP UTILITIES
:h1 res=2003. "Group - Utilities"
:p.This best utility is the control panel. If you get a new printer for
your PC you will probably need to install printer drivers through the
:hp2.Control Panel:ehp2.. If you have a program that needs fonts installed
this is also the best place.
:p.Misellaneous hard disk programs may be found here as well. If any
questions on using these things arises I would point you to the OS/2
Command Reference for online help. Else try the OS/2 EE Volume 1 manual.
.*
.* HELP_CONTROL_PANEL
:h1 res=2006. "Control Panel"
:p.This OS/2 PM utility allows you to personalize your PS/2 workstation.
Possible settings to configure are:
:eol.
:li.System colors
:li.Window size borders
:li.Country specific information
:li.Time and date
:eol.
:euserdoc.

```

Figure 4. Is an Example of IPF Help Panel Source Code

References

Dynamic Link Libraries

1. *Building Programs*, "OS/2 Programming Tools and Information Version 1.2", pp. 5-1 to 5-5.
2. *Programming Guide*, "OS/2 Programming Tools and Information Version 1.2", pp. 34-20 to 34-21.

Hooks

3. *Presentation Manager Programming Reference Vol 2*, "OS/2 Programming Tools and Information Version 1.2", pp. 10-4 to 10-18.

Help Manager

4. *Programming Guide*, "OS/2 Programming Tools and Information Version 1.2", pp. 15-1 to 17-23.
5. Enright, Valerie, Personal Systems Developer "Online Help for PM Applications", Winter 1991, pp. 57-63.

OS/2 PM

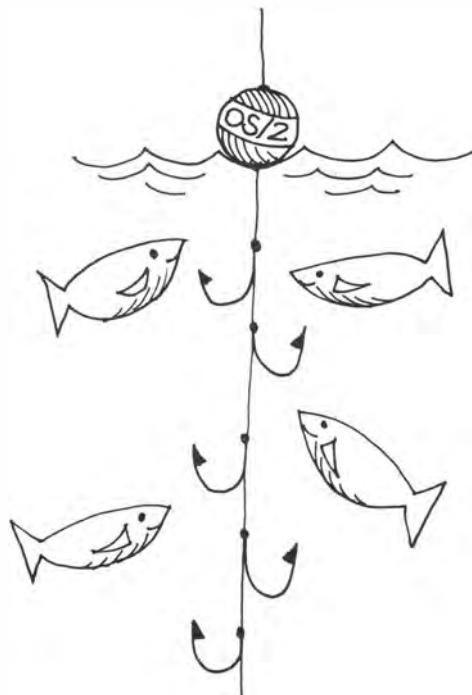
6. Cheatham, Reich, and Robinson, *Personal Systems Developer*, "Presentation Manager Architecture", February 1989, pp. 33-43

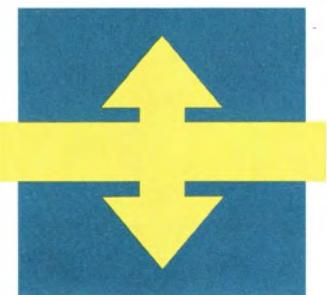
Figure 4 is a simple example of IPF help panel source code.

Programming system hooks is hardly a clear and obvious subject to be sure. Hopefully this article will serve as future inspiration. Since it has hardcoded examples it is not truly universal in the design.

The application described in this article should give a good example of the power and flexibility of OS/2 PM. Impressions of PM as a slow, restrictive, under mandated windowing system should be dispelled. PM is instead a strongly defined environment in which developers may program logical expectations.

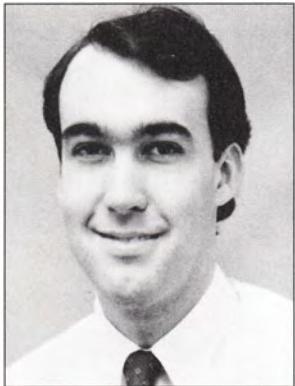
Ken Tabor, Software Usability, IBM, 5 West Kirkwood Ave, Roanoke, Tx 76299. Mr. Tabor is working in the Office Vision Human Factors department at the Westlake Programming Lab. His work includes interactive prototyping theory and advanced interface design. He is currently enrolled in IBM's cooperative work-experience program and working on his BA in Computer Science at the University of North Texas.



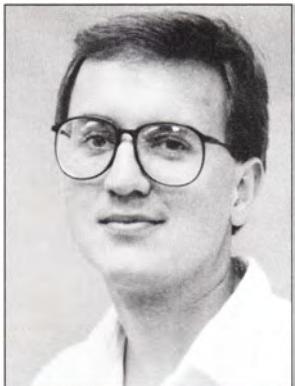


Application Enablers

IBM SAA Networking Services / 2



Timothy S. Huntley



Steven T. Joyce

by Timothy S. Huntley and Steven T. Joyce

This is the first of two articles about IBM's new SAA Networking Services/2 product. In this article we'll learn just what it can do for OS/2 application developers. In the next article, we'll hear from one of the developers who beta-tested the product for IBM.

IBM SAA Networking Services/2 is an exciting new addition to the OS/2 Extended Edition environment for peer-to-peer communications. It provides significantly improved performance, configuration, usability, tools and sample programs for Advanced Program-to-Program Communication (APPC). SAA Networking Services/2 also adds new support for Advanced Peer-to-Peer Networking (APPN) and the Common Programming Interface for Communications (CPI-C). This new technology, combined with other recently announced products, makes OS/2 EE a powerful cooperative-processing system. (For further information on APPC refer to the following references listed at the end of this article: Berson, Henderson, pp. 289-312 and Walker, pp. 313-324.)

During the past several years, a new computing paradigm has begun to form. It goes by names like client-server computing, distributed processing, or cooperative processing. Historically, a terminal was attached to a large "host" computer that did all the processing. The new paradigm is based on using communications between more than one computer to share a task. The main reason for the change was the enormous increase in the power of small computers. Ideally an application will take advantage of each type of processing available. In general, workstations provide excellent input and output. Mainframe computers provide fast processing and large amounts of shared disk storage.

As people begin to develop new cooperative applications, they are discovering the strength of APPC. It is available on every major IBM operating system and dozens of non-IBM platforms. It is a crisply defined, flexible, open architecture. APPC offers advanced features, such as security, confirmation flows and error handling, and works the same way in both LAN and WAN environments.

SNA BACKGROUND

In order to understand what Networking Services/2 does, APPC, APPN and CPI-C must be understood. An easy comparison can be made with the way the phone is used. First, the person to whom a message needs to be related is determined, the phone is dialed and a connection is made. After the needed information is exchanged a definite end to the conversation is made by saying good-bye. APPC provides the same functions between application programs. The application tells APPC who it needs a conversation with. APPC starts the conversation and then allows the programs to begin exchanging data. Like the phone system, APPC is half-duplex. This means that programs can only communicate one way at a time. When all the data has been exchanged, the programs end the conversation.

If APPC can be compared to the use of a phone, the function of APPN can be compared to the phone company. APPN allows APPC and the applications to communicate between rooms or between states. APPN handles communication in the same way the phone network handles routing, through trunks, switches and branches all invisible to the callers. So, communication can take place without requiring APPC or the application to do anything differently.

CPI-C does not have a parallel to the phone comparison. It is a single, common interface that all applications can use on all platforms. The interface CPI-C provides is like a single, international language spoken on the phone and understood by everyone.

PACKAGING

Networking Services/2 is an IBM product available since March, 1991. The package includes the following:

- 3.5" and 5.25" Program Diskettes
- Networking Services/2 Installation and Network Administrator's Guide
- Networking Services/2 System Management Programming Reference
- Networking Services/2 APPC Programming Reference
- Networking Services/2 Problem Determination Guide

Networking Services/2 requires that OS/2 Extended Edition Version 1.2 or 1.3 already be installed. As shown in Figure 1, NS/2 replaces the APPC functions provided by OS/2 EE's Communications Manager at the same time all new functions, tools and samples are added.

With the announcement of Networking Services/2, IBM made a statement of direction to let customers know that the product would be integrated into the OS/2 Communications Manager. Networking Services/2 and the OS/2 Communications Manager were developed as separate products in order to give users the use of Networking Services/2's state-of-the-art networking technology even before the integrated release of OS/2 EE.

PERFORMANCE

APPC performance using Networking Services/2 has been significantly improved allowing both new and existing applications to realize substantial gains. Not only are performance times improved, CPU utilization is reduced giving more CPU cycles back to the applications. Using two PS/2 model 80's with 20 MHZ clocks and 8MB of memory, benchmarks ran anywhere from 2 to 4 times faster using Networking Services/2 than with

OS/2 EE 1.2. On a 4Mb Token Ring LAN, setting up an APPC conversation and sending 100 bytes of data took 15.3 milliseconds using OS/2 EE 1.2. The same transaction ran in 7.2ms with Networking Services/2.

Transferring 100,000 bytes of data took .524 seconds to run with OS/2 EE 1.2 and .233 seconds using Networking Services/2. This is almost the full capacity of the LAN, using only two machines. This improvement allows APPC to perform competitively with NetBIOS for small data transfers and markedly faster for large data transfers.

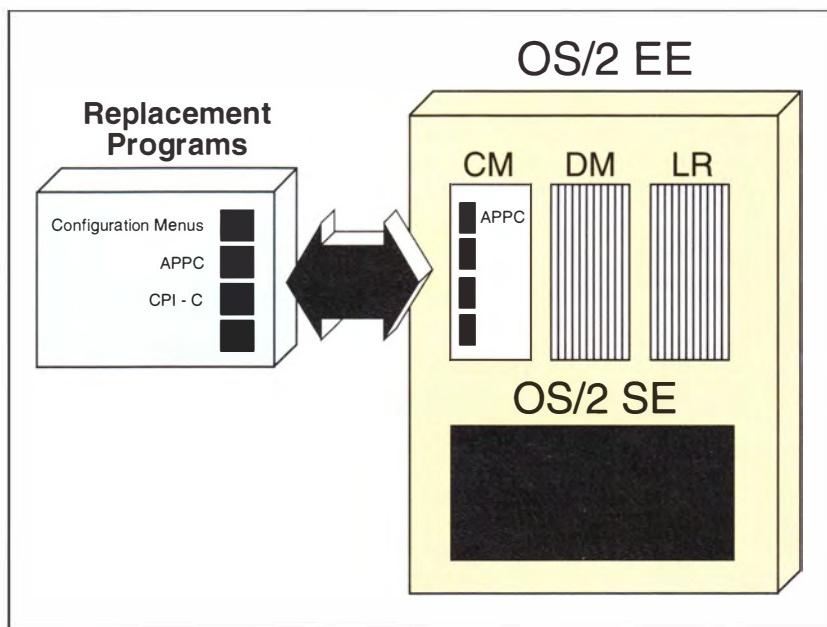


Figure 1. Replacement Programs

CONFIGURATION

Our design philosophy for Networking Services/2 was to remove as much of the configuration detail as possible. For advanced users, the ability to manipulate any of the configuration parameters was still a requirement. The problem the Networking Services/2 development team had to overcome was how to make configuration both simple for new users and comprehensive for advanced users. We developed a three-part strategy to make this possible.

APPN is the first part of this strategy. APPN allows each computer in the network to be either a network node or an end node. Here are the roles each plays (only the first two require user definition for network or end nodes).



SAA Networking Services/2 provides improved performance for applications using APPC, APPN, and CPI-C

End Nodes

- Define network names (Logical Unit names) for that machine
- Define a link to a network node
- Register the network names with its network node
- Ask the network node to find other LUs in the network
- Ask the network node to determine routes through the network
- Tell the network node what other links it has into the network
- Pass management services alerts to the network node

Network Nodes

- Define network names for that machine
- Define links to other network nodes
- Keep track of end nodes it supports and what network names they have
- Talk with other network nodes to:
 - Keep track of the network's topology.
 - Find LUs in the network.
 - Reply when other network nodes are looking for LUs.
- Calculate routes through the network
- Route traffic
- Pass network management alerts to a network management focal point

Most workstations will be end nodes. All their users need to do is define what they have on their workstation. When an application needs to start a conversation with another computer in the network, Networking Services/2 will ask a network node to find the communications partner and then start communications. The dynamic searches and routing are all invisible to the application.

APPN is currently offered on the IBM AS/400 and System/36. Support for APPN has also been announced for DPPX/370 and the 3174 Establishment Controller. In addition, Apple,

Novell, Siemens-Nixdorf, and SSI have announced their intention to market products serving as APPN end nodes.

The second part of the Networking Services/2 strategy is to default or discover everything possible. All APPN products are shipped with architected modes and classes of service including batch and interactive types, both secure and non-secure. These predefined modes should handle almost all situations. With Networking Services/2 it is no longer necessary to define the name of every communication partner. Implicit names and definitions can be created when sessions are started. Additionally, session limits can be implicitly initialized. Finally, defaults for transaction program (TP) names, operation, and type can be specified, removing the need to configure TP definitions for application programs.

The third piece not only minimizes definitions, it also makes the definition easier. With its Presentation Manager interface, Networking Service/2 provides a new user with an easy method of configuration. When a configuration is complete, the configuration file is stored in a text file. This file, referred to as the node definition file, shows each verb defined during the configuration. Figure 2 is an example node definition containing only the minimum parameters needed to run as an end node in a LAN environment.

A more-experienced user can edit this file and make changes to any of the parameters. Since the node definitions file can be manipulated by a simple text editor, users can easily build and manage large numbers of configuration files. A central administrator can build and maintain node definition files for each machine by modifying a few simple parameters.

Existing OS/2 EE configurations are automatically migrated to the new formats when Networking Services/2 is installed. Information on installing and configuring any OS/2 workstation for Networking Services/2 is described in the *IBM SAA Networking Services/2 Installation and Network Administrator's Guide*.

DEVELOPMENT ENVIRONMENT

Improvements for programmers who design, code, test, and debug APPC applications were considered essential elements of Networking Services/2. The improvements include the new CPI-C interface, new software tools and sample programs, the ability to develop APPC applications on a single machine, and new access to management services functions.

COMMON PROGRAMMING INTERFACE-COMMUNICATIONS

When APPC was first designed, the line flows were crisply defined. The programming interface functions were defined but the semantics were not. Each product ended up with variations in how APPC was invoked. Because people developing cooperative applications are frequently working on multiple platforms, a consistent interface has become very important. CPI-C is a *call interface* that is common across different operating systems and programming languages and is evolving as the transaction programming interface for both SNA and OSI. Future applications can be developed without concern for the type of network they will run on. This interface is available for NS/2, VM, MVS, CICS, AS/400, IMS, DPPX/370, 4680 Store Systems, and Series/1. Since the API has been given to the X/Open consortium and licensed to several other companies, it is not an IBM-only solution.

TOOLS

We surveyed current OS/2 APPC users to find out what tools they needed the most. Based on that feedback, we have provided several interesting programs. The first is a trace formatter. Rather than looking at hex dumps and offsets, the programmer can see English text describing verbs and line flows. There is also a tool that will allow starting, stopping or filing APPC traces from the command line.

OS/2 EE provides powerful verbs to find out what is defined and active in the system. A new program, PMDSPLAY, allows easy information selection and writes it to the screen or to a file. It also runs as an APPC application, so you can locally examine this information on other machines.

SAMPLES

Sample programs are often the easiest way for a programmer to get started developing in a new environment. Networking Services/2 provides many excellent examples, written in C, COBOL and REXX. Included are: a file transfer program, a simple messaging program, a configuration program, a system management program, examples showing how to use the new CPI-C interface, and the source for the PMDSPLAY program mentioned earlier.

Local-Remote Transparency

Another important function offered to developers is the ability to run both sides of an application on a single machine. The difference between a partner being located on a remote machine or the same is transparent to the application and requires no special configuration. This allows developers to do testing in a more convenient, controllable environment. It also allows systems that are providing server applications to use client programs without making special considerations.



Management Services

Networking Services/2 provides additional support for developers wishing to write management services (MS) applications. The programming support consists of several verbs that allow a program to register or unregister itself as a management services application and for sending MS data. By registering its name, an application can



```

DEFINE_LOCAL_CP   FQ_CPN_NAME(USIBMSER.USER1)
                  CP_ALIAS(MYCP);

DEFINE_CONNECTION_NETWORK  FQ_CN_NAME(APPN.CONNET1)
                           ADAPTER_INFO(
                           DLC_NAME(TBMTRNFT)
                           ADAPTER_NUMBER(0));

DEFINE_LOGICAL_LINK    LINK_NAME(SERVER)
                           DLC_NAME(IBMTRNET)
                           ADAPTER_NUMBER(0);

DESTINATION_ADDRESS(X'100005A6D000')
                           ACTIVATE_AT_STARTUP(YES)
                           CP_CP_SESSION_SUPPORT(YES);

DEFINE_DEFAULTS   DIRECTORY_FOR_INBOUND_ATTACHES
(C:\CMLIB\APPN);

START_ATTACH_MANAGER;

```

Figure 2. Sample Node Definitions File



receive MS data in the new MDS_MU format as opposed to the NMVT format. Programs still using the NMVT format will be able to continue to use this format with no change to the application.

APPLICATIONS

The APPC programming interfaces offered on OS/2 EE Versions 1.2 and 1.3 are supported on Networking Services/2. This means that any program that uses OS/2's APPC today can run, without change, and will have enhanced performance after NS/2 is installed.

Some of the IBM programs that currently run APPC are:

- OfficeVision
- OS/2 Database Manager
- Distributed Console Access Facility (DCAF)
- LAN-to-LAN Wide Area Network Program
- AS/400 PC Support Program

SUMMARY

Networking Services/2 is a major step towards providing a base for distributed processing. Because of its excellent performance, easy configuration and support of APPC's advanced functions and APIs, the development of NS/2 marks a significant advancement in bringing state-of-the-art networking to user desktops.

Acknowledgements

We appreciate the careful review and many helpful comments offered by two of our IBM associates, Rick McGee and John Q. Walker.

References

Berson, Alex. *APPN: Introduction to LU6.2*. McGraw-Hill Publishing, 1990. (ISBN 0-07-005075-9)

Henderson, Sam and Jim Kovaric. "Advanced Program-to-Program Communications (APPN)". *OS/2 Notebook: The Best of the IBM Personal Systems Developer*, Microsoft Press, 1990. (ISBN 1-55615-316-3, G362-0003).

IBM SAA Networking Services/2 Installation and Network Administrator's Guide. IBM document number SC52-1110.

Pozefsky, Diane P., Daniel A. Pitt and James P. Gray. "IBM's Systems Network Architecture" reprinted: *Computer Network Architectures and Protocols*, 2nd ed. Carl A. Sunshine, editor, New York: Plenum Publishing Corporation. 1989.

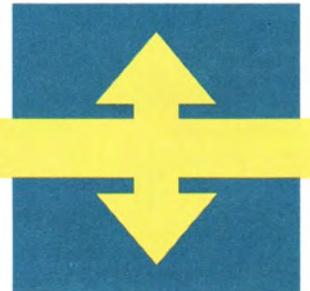
Walker II, John Q. "The APPN Attach Manager" *OS/2 Notebook: The Best of the IBM Personal Systems Developer*, Microsoft Press, 1990. (ISBN 1-55615-316-3, G362-0003).

Tim Huntley, IBM Corporation, Department E42, Building 673, P.O. Box 12195, Research Triangle Park, NC 27709. Mr. Huntley is an Associate Programmer in Architecture and Telecommunications. He was a member of the IBM SAA Networking Services/2 development team and is currently working with vendors that are writing APPN applications. He joined IBM in 1989 in RTP where he was involved with APPN architecture. Mr. Huntley received a BS in Computer Science from North Carolina State University.

Steven T. Joyce, IBM Corporation, E42/673, P.O. Box 12195, Research Triangle Park, NC 27709, (919) 254-4465. Mr. Joyce just completed managing the planning and testing of IBM SAA Networking Services/2. He is now a member of the APPN Market Enablement department in the Architecture and Telecommunications organization. He joined IBM in Raleigh, North Carolina in 1983, where he was involved with the quality assurance and testing of IBM's office systems. Mr. Joyce received a BS in Computer Science from North Carolina State University.

Application Enablers

Multiple Conversation OS/2 Server Using IBM SAA



by Robert Lindsay

American Management Systems, Inc., is a 21-year-old company applying information technology to solve complex management problems. As an early user of OS/2 Extended Edition, AMS has pioneered the development of transaction-based PC applications. As noted in the preceding article by Huntley and Joyce, Networking Services/2 provides a powerful communications capability in this domain.

Recently, AMS had the opportunity to utilize a new OS/2 EE product, IBM® SAA Networking Services/2™. Throughout this article, Robert Lindsay, senior principal, will report AMS's experience with this product, including their perception of how this extends the usefulness of OS/2 in the transaction processing domain.

The AMS MicroTradeLine™ system was originally written as a banking mainframe application to process international letters of credit and collections. It was designed and written during the late 1980's and is based on a transaction-oriented architecture. AMS has written many different transaction systems and has refined an architectural approach that maximizes application independence across system platforms.

Two years ago we started to address the migration of this application onto a LAN-based environment for several reasons:

- CICS™ regions are expensive mainframe resources, and many of our CICS customers felt the migration of their applications to a LAN would result in lower operating costs.
- Migration to a LAN implied the ability to distribute application workload across multiple processors and achieve better control of throughput.

- The intense push to adopt SAA™ as a standard was introducing LAN-based hardware and workstations.
- The ability to operate in a distributed manner utilizing LANs would allow some customers to decentralize and achieve more efficient operations at the regional and field level.

APPROACH

Because mainframe systems are designed to support hundreds of terminals, we felt communication between a LAN version of the application and the mainframe version of the application would be required for almost all customers. This would allow phased integration of the LAN-based system into a mainframe environment and would also facilitate interaction between our product and other customer mainframe-based systems. Additionally, because this project would become a new standard for future LAN-based systems, we wanted the most robust and flexible LAN environment that could be obtained.

For these reasons, we selected IBM PS/2's™ running OS/2 EE™ as our LAN server platform. Eventually we plan to support both OS/2 and DOS Windows™, but the Windows support has been deferred until later.

To maintain compatibility with mainframes, MicroFocus COBOL/2™ (essentially IBM COBOL/2 with ANIMATOR™ support) is used as the main language in the LAN version of this application. The OS/2 EE Database Manager is used for SQL database services. During the development project, presentation services have been provided by C programs that support 3270 screen activities in a workstation environment.



Robert Lindsay



These are being replaced in the final product by a graphical interface (GUI) written in C. Finally, for communications, we chose CPI-C.

Early in the project, modelling of alternative designs indicated that database access would be a critical factor in achieving performance goals. Benchmark data indicated the OS/2 EE database manager had acceptable data access time. However, because of the large number of application tables (more than 200), we were unable to use Remote Data Services. When we attempted to use APPC to create a client/server based access, long turnaround times were encountered. Finally, we considered NetBIOS or possibly TCP/IP, either of which would mean a messier design, and a compromise with our goal of having an all-SAA implementation.

Fortunately, IBM SAA Networking Services/2 arrived and provided us with an implementation of CPI-C, which is better for us than APPC. What's more, it does so in an efficient and elegant product.

HOW IBM SAA NETWORKING SERVICES/2 WORKS

IBM SAA Networking Services/2 provides an upgraded and more efficient replacement of the Communications Manager APPC API — an advantage even for programmers not

interested in CPI-C or APPN, because IBM SAA Networking Services/2 makes APPC applications execute faster. In addition, IBM SAA Networking Services/2 provides an implementation of the SAA CPI-C (see the IBM SAA Common Programming Interface Communications Reference SC26-4399).

As an added advantage, IBM SAA Networking Services/2 provides on-line utilities to dynamically define most of the network configuration. Programmers who have struggled with setting up an APPC network under OS/2 can see this last advantage of IBM SAA Networking Services/2 immediately. We found that IBM SAA Networking Services/2 actually works better than advertised and solves not only design issues, but handles many big development problems too.

Best of all, IBM SAA Networking Services/2 creates the CPI-C interface as an additional layer over APPC, allowing SAA CPI-C programs to interface with existing systems such as CICS (which supports its own version of LU6.2) without a lot of hassle. Considering these features, and since SAA compliance was our goal, IBM SAA Networking Services/2 was our only vehicle short of writing our own CPI-C interface.

OVERALL ARCHITECTURE

Because of the database size and transaction activity in this application system, multiple database servers and multiple transaction servers are needed to implement the LAN version. Each transaction server handles from five to 10 workstations, and each database server handles up to 10 transaction servers. A portion of our application deals with textual data, and a special text server was constructed for just the storage, retrieval and processing of application text (see Figure 1).

All of these servers use CPI-C. This has proven to be an effective, simple protocol for the construction of transaction-oriented programs. For maximum SAA compatibility, in our application both sides of each conversation use CPI-C. The only exception is the link to the mainframe which utilizes CICS LU6.2. We would have preferred to use CPI-C for the mainframe server, but many of our clients run CICS and do not yet have the current version of MVS to support CPI-C.

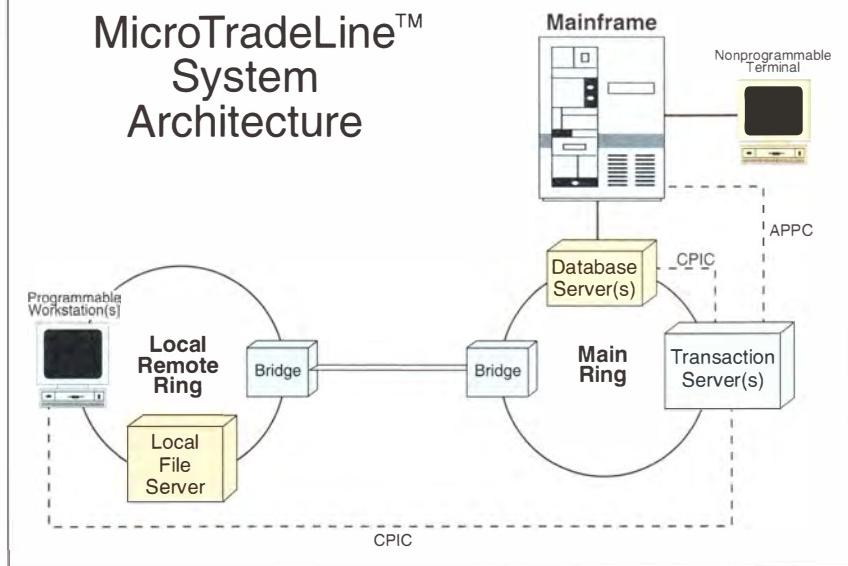


Figure 2. OS/2 Facilitates Multiple-Conversation C-PIC Servers

WHY BUILD CPI-C SERVERS?

There are several advantages built into CPI-C, and one disadvantage which can be overcome through the facilities of OS/2. The chief advantages are a simplified set of verbs, and the ability of the communications software to maximize throughput when CPI-C is used; the disadvantage is architectural and occurs because CPI-C specifies that a process can RECEIVE only a single conversation at a time. (APPC allows a receiving process to handle multiple conversations.) Fortunately, OS/2 provides a rich set of multitasking facilities which allow us to construct a CPI-C server solution that handles multiple conversations.

Why construct a server that handles multiple conversations? Our design goal was at least 100 end user workstations. To keep our LAN-based solution manageable we wanted to have 10 or fewer server machines. We felt that more servers would generate operational complexity and endanger system reliability.

As an additional constraint, to preserve database integrity, we wished to have a client and a server remain in conversation for at least a unit-of-work as defined by the application. (This would be from the initiation of a conversation or a SYNCPOINT until the next SYNCPOINT.) And we wanted conversations to exist for the life of the client, so we don't spend response time ALLOCATING conversations.

If we used the architected CPI-C model, we would need 100 active conversations, each to a separate 500 kb transaction server program (we're using COBOL). This would require about eight transaction server machines to process the application logic. Each transaction server program would need access to all the application tables, so we would need 100 database server programs for each portion of the database. Because the database portions cannot be split across systems (for integrity) we could only support a database server program large enough for two SQL tables per machine. Since we have about 200 application tables, we would need 100 database machines!

This would clearly be a wasteful design. We knew from measurements that we could expect only 10 simultaneously active units-of-work from 100 workstations. If we could

share database server programs we could reduce the number of required machines to our goal of 10. But we didn't want to lose the advantages of CPI-C for our conversations. The only solution was to construct OS/2 based servers capable of handling multiple CPI-C conversations per server program.



OS/2 BASED CPI-C SERVER DESIGN

Our OS/2-based solution starts a separate small communications process for each client conversation. These conversation processes queue up for the use of a server via OS/2 queue mechanisms. At the same time, available servers queue up to be assigned to a conversation. Finally, a single process reads both queues and assigns conversations to servers for the duration of a unit-of-work. Once assigned, both the server and the communications process remain dedicated to each other until the end of a unit-of-work, then the server and the communications process disconnect from each other. On the next request from his client, the conversation process will again join the request queue. At the end of a unit-of-work, the server immediately becomes a free-agent and can be assigned to the next waiting client conversation. (See Figure 2.)



A FEW DESIGN DETAILS

One of the efficient aspects of this design is the minimum amount of CPU resource required to handle each request as it's passed from a client to a server. IBM SAA Networking Services/2 does a great job at getting data transported at little cost. This would be wasted if the OS/2 mechanisms used to handle the data were lengthy. So we made extensive use of semaphores, DosMuxSemWait, and shared buffers in creating the server design.

A fast-safe (FS) RAM semaphore and a NAMEd, shared storage area are used by the anchoring process, and form the initial way in which the conversation process and the server process make themselves available. The NAMEd, shared memory area holds additional semaphores which are used during the actual transfer of requests.

The server handles multiple conversations



DosMuxSemWaits are used extensively because of two key features: they provide edge triggering; and they allow a timeout value. This means that monitoring for server failure does not require additional threads, since the semaphore will timeout if not cleared within the allowed design limits.

Finally, shared buffers are required for interfacing to IBM SAA Networking Services/2. These buffers are allocated in the conversation processes and shared with a server process at the time of assignment. So the data is never moved inside of the OS/2 server, despite being handled by multiple processes.

In our server, the OS/2 APIs are being accessed from both C and COBOL programs and work perfectly well despite the very different nature of the run-time environments of these languages.

THE IBM SAA NETWORKING SERVICES/2 DEVELOPMENT ENVIRONMENT

Prior to IBM SAA Networking Services/2, the construction of a LAN client-server application required several machines, and rebooting these machines whenever the configuration changed. This meant a more complicated development environment and therefore more risk.

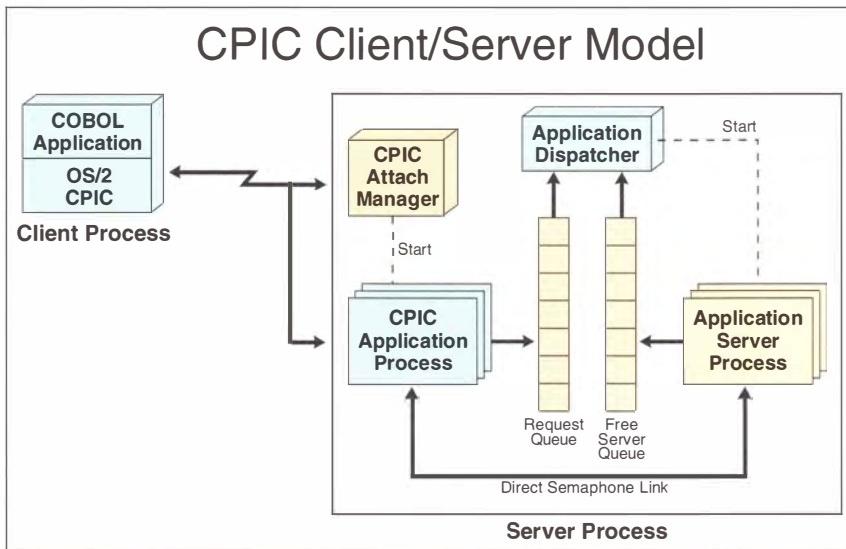


Figure 2. OS/2 Facilities Multiple-Conversation CPI-C Servers

IBM SAA Networking Services/2 has changed this through two key features: the ability to define a partner LU on the same machine; and the ability to dynamically change the parameters, location, and names of transaction programs.

The first key feature allows us to develop and debug both sides of a conversation on the same machine. Not only does this make it easier to setup the same version of system for both the client and the server, it also gives us a common trace log of the communications. The sequence of messages is immediately and clearly visible.

Being able to execute both LUs on the same system also provides us with the ability to see the interactive debugging tools for both sides of the conversation on the same desktop. We can now step though both sides of the conversation, seeing the variables used in each program, without ever leaving our screen.

The second key feature allows us to turn on and off debugging tools like CodeView® for attached programs without requiring a reboot due to configuration changes. A great deal of debugging time is spent in setting up the trap and trace conditions to isolate a communications problem. Forcing reestablishment of these conditions because of a configuration change reboot was eliminated for us by IBM SAA Networking Services/2.

With IBM SAA Networking Services/2, if an attached program is failing, the programmer can dynamically alter the name and parameters being attached to execute CodeView (for a C program) or ANIMATOR (for a COBOL program) instead of the transaction program.

Another handy feature of IBM SAA Networking Services/2 is the use of the ASCII-based network definition file (NDF). When we need to share our configuration with another site (or programmer) we can send them the NDF, and they can integrate it with their own NDF using their favorite text editor.

The final debugging feature we like is the FMTTRACE command. We have even used this to determine the contents of the SNA messages being sent to us by the mainframe (try doing that with hex dumps sometime).

All this adds up to less risk, more predictable development target dates, and fewer midnight sessions trying to figure out who sent what, when.

MEASURED RESULTS

Even a great development tool looks pretty shabby if the final system doesn't perform well. To validate our IBM SAA Networking Services/2 OS/2 CPI-C server design, several benchmarks were performed (see table, Figure 3). These established the time required to access data without using IBM SAA Networking Services/2 and our OS/2 server design, plus the time to access data locally (in the same machine) and across a network using IBM SAA Networking Services/2 and our OS/2 server design. Finally, access to data on the mainframe was benchmarked for comparison.

The results are strongly encouraging, and demonstrate how OS/2 and IBM SAA Networking Services/2 can work together to provide an efficient multiple conversation CPI-C server platform.

The first access time of 61 ms is the access time required to request and return a data record locally (without the benefit of our server overhead or IBM SAA Networking Services/2) on a fast 386 server machine. The second number of 74 ms is the time to access the same record through our client-server programs via IBM SAA Networking Services/2 in the same server machine. We can see that we have added 13 ms of overhead somewhere between IBM SAA Networking Services/2 and our use of OS/2 features.

The third number is the same access, now using a network to transport the request and return the data. Our requestor machine is a little slower than our server machine. Still, we see that using a network has cost us only 3 ms over the time required to access the data within the server machine.

If we didn't have the fast server machine with IBM SAA Networking Services/2 and just accessed that data locally on our slower requestor machine we would get the fourth benchmark of 83 ms. We can conclude our entire OS/2 overhead, our network transport time and our IBM SAA Networking Services/2 overhead is less than the additional time to access the data locally on a slower machine. Thus our measured results validate the use of a client-server architecture.

The fifth line measures our ability to access data on the mainframe. The time required to ship the request via an existing conversation to CICS and receive back the data is 250 ms. This is not a lot of time as long as we can satisfy most of our requests from the LAN environment and only have to occasionally access the mainframe.



Measured I/O Server Results

• Local 25 Mhz 386 access with 17 ms drive	= 61 ms
• CPIC local 25 Mhz 386 access 17 ms drive	= 74 ms
• CPIC network access 16 Mhz 386 → 25 Mhz 386	= 77 ms
• Local 16 Mhz 386 access with 25 ms drive	= 83 ms
• Network access to CICS LU6.2 server	= 250 ms

Figure 3. Measured I/O Server Results

CONCLUSION

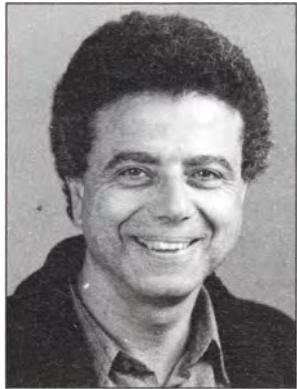
This article has attempted to relate some of our experience with the IBM SAA Networking Services/2 communications product and OS/2. We feel that these products provide an excellent platform for the development and execution of servers in a LAN environment.

Robert Lindsay, American Management Systems, Inc., 1777 North Kent St., Arlington VA 22209, is an AMS senior principal, and provides design and advanced development support for AMS. Mr. Lindsay produced many key portions of AMS's CORE foundation software. Prior to joining AMS, he developed and marketed a COBOL compiler for the IBM Series/1® and the IBM PC®. Mr. Lindsay holds a BS degree in Electronic Engineering from the Massachusetts Institute of Technology.

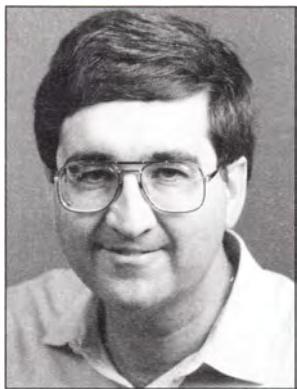


Extended Edition

Client-Server Solutions: OS/2 or Windows 3.0 ?



Robert Orfali



Dan Harkey

by Robert Orfali and Dan Harkey

From Client-Server Programming with OS/2 Extended Edition, Robert Orfali and Dan Harkey, ©1991 by Van Nostrand Reinhold. (Adapted by permission of the Publisher. All rights reserved.) This book may be ordered from IBM Mechanicsburg as publication number G325-0650.

Client-Server computing promises a great deal: industrial strength applications provided by inexpensive, network attached PCs. For the users of PCs, it promises to maintain the autonomy and ease of use they have enjoyed, while overcoming the stand-alone isolation of PC software. But, client-server computing adds new demands on PC operating systems. Client-Server is primarily a relationship between programs running on separate machines¹. As such, it requires an infrastructure to do things standalone PCs never had to worry about such as preemptive multitasking, interprocess communications and robust peer-to-peer communications over LANs which extend the interprocess communications services.

This article examines the question: can the traditional MS-DOS operating system, augmented with Windows 3.0, support these new demands, or do we need to move on to OS/2? To answer this question we will study the requirements of the client and server separately. Their functions are very different and may produce separate answers. For each we ask: what does it do? What does it need from an operating system? We then match the capabilities of OS/2-PM and DOS-Windows to the requirements to determine which is the best fit.

THE ANATOMY OF A SERVER PROGRAM

The role of a server program is to serve multiple clients who have an interest in a shared resource owned by the server. The following describes what a typical server program does:

- **WAITS FOR CLIENT-INITIATED REQUESTS.** The server program spends most of its time passively waiting on client requests, in the form of messages, to arrive over a communication session. Some servers assign a dedicated session to every client. Others create a dynamic pool of reusable sessions. And, some provide a mix of the two environments. Of course, to be successful the server must always be responsive to its clients and be prepared for *rush hour traffic* when many clients will request services at the same time.
- **EXECUTES MANY REQUESTS AT THE SAME TIME.** The server program must do the work requested by the client promptly. Clearly a client should not have to depend on a single-threaded server process. A program that does will run the risk of having a client hog all the system's resources and starve out its fellow clients. The server must be able to concurrently service multiple clients while protecting the integrity of shared resources (see Figure 1).
- **TAKES CARE OF VIP CLIENTS FIRST.** The server program must be able to provide different levels of service priority to its clients. For example, a server can service a request for a report or batch job in low priority while maintaining OLTP-type responsiveness for high priority clients.



- **INITIATES AND RUNS BACKGROUND TASK ACTIVITY.**

The server program must be able to run background tasks that are triggered to perform chores that are unrelated to the main program's thrust. For example, it can trigger a task to download records from a host database during non-peak hours.

- **KEEPS RUNNING.** The server program is typically a mission-critical application. If the server goes down it impacts all the clients that depend on its services. The server program and the environment in which it runs must be very robust.

- **GROWS BIGGER AND FATTER.**

The server program must be scalable and modular. The server program (and the resources it controls) should be able to run in parallel on several machines.

Here's a short list of the multitasking services that servers need from their operating system:

- **Task Preemption.** An operating system with preemptive multitasking allots fixed time-slots of execution to each task. Without preemptive multitasking, a task must voluntarily agree to give up the processor before another task can run. It is much safer and easier on the server application developers to place the operating system in charge of task switching.

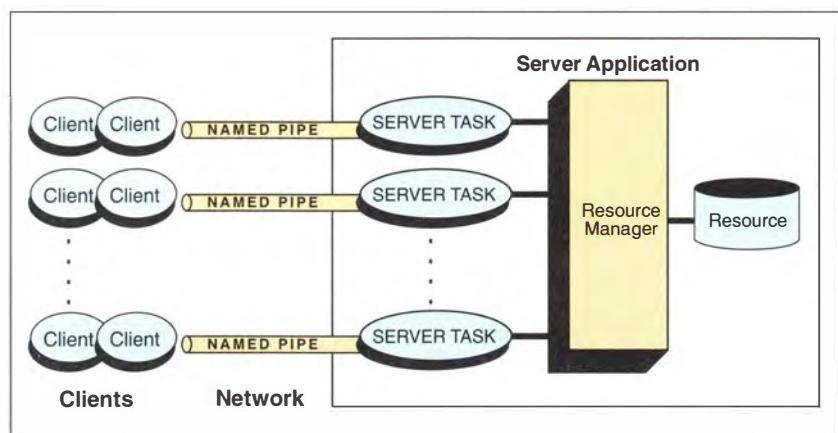


Figure 1. The Anatomy of a Server Program

SO WHAT DOES A SERVER PROGRAM NEED FROM AN OPERATING SYSTEM?

It should be apparent from the description above that server programs exhibit a high level of concurrency. Ideally, a separate task (i.e., concurrently executing unit of code) will be assigned to each of the clients the server is designed to concurrently support. Task management is best done by a multitasking operating system. Multitasking is the natural way to simplify the coding of complex applications that can be divided into a collection of discrete and logically distinct, concurrent tasks. It improves the performance, throughput, modularity, and responsiveness of server programs. Multitasking also implies the existence of mechanisms for intertask coordination and information exchanges.

Servers also require a high-level of concurrency within a single program. Server code will run more efficiently if tasks are allocated to parts of the same program rather than to separate programs (these tasks are called co-routines or threads). Tasks within the same program are faster to create, faster to context switch, and have easier access to shared information.

- **Task Priority.** An operating system must dispatch tasks based on their priority. This feature allows servers to differentiate the level of service based on their client's priority.

- **Semaphores.** An operating system must provide simple synchronization mechanisms for keeping concurrent tasks from bumping into one another when accessing shared resources. These mechanisms, known as semaphores, are used to synchronize the actions of independent server tasks and wake them up when some significant event occurs.

- **Interprocess Communications.** An operating system must provide the mechanisms that allow independent processes to exchange and share data.

Servers require a high level of concurrency



FEATURE	DOS-WINDOWS	OS/2-PM
TASK PREEMPTION	Only for DOS (Windows programs are non-preemptive).	All OS/2 tasks (PM programs are cooperative using a single system queue)
TASK PRIORITIES	Only for DOS programs (user assigned).	For all tasks.
SEMAPHORES	No	Yes
INTERPROCESS COMMUNICATIONS		
DDE	Yes	Yes
QUEUES	No	Yes
SHARED MEMORY	No	Yes
NAMED PIPES	No	Yes
LOCAL/REMOTE TRANSPARENCY	Remote only, with Named Pipes.	Local or remote with Named Pipes.
THREADS	No	Yes
INTERTASK PROTECTION	Poor. All Windows applications share the same Local Descriptor Table (LDT). No hardware enforcement of protection.	Extensive. Hardware enforced protection between applications.
REENTRANT OS CALLS	Windows only (DOS not re-entrant).	Yes
OS ROBUSTNESS	Poor. System runs under real-mode DOS. One application can easily crash the entire system and bring down all the tasks.	Extensive. System and programs run in protected mode. Four levels of privilege are supported by hardware.
FILE SYSTEM	FAT only (Share mode supported)	HPFS and FAT (Share mode supported)

Table 1. Server Platform: Comparing DOS-Windows and OS/2 PM



- **Local/Remote Interprocess Communications.**

An operating system must allow the transparent redirection of interprocess calls to a remote process over a network without the application being aware of it. The extension of the interprocess communications across machine boundaries is key to the development of applications where resources and processes can be easily moved across machines (i.e., they allow servers to grow bigger and fatter).

- **Threads.** These are units of concurrency provided within the program itself.

Threads are used to create very concurrent event-driven server programs. Each waiting event can be assigned to a thread that blocks until the event occurs. In the meantime, other threads can use the CPU's cycles productively to perform useful work.

- **Intertask Protection.** The operating system must protect tasks from interfering with each other's resources (such as memory segments). A single task should not be allowed to bring down the entire system. Protection also extends to the file system and calls to the operating system.

THE SERVER PLATFORM: OS/2 OR WINDOWS?

Now that we know what a server platform requires from an operating system, Table 1 summarizes what DOS-Windows and OS/2-PM each have to offer.

The table shows that OS/2 is clearly on top as a server platform. OS/2 provides a very granular level of multitasking which is optimized for concurrency on a single-user machine. The architecture builds on a three-tiered tasking model: the thread, the process, and the session (or screen group). Threads run within processes which in turn

run within screen groups. Figure 2 shows the relationships in the hierarchy. The thread is the basic unit of concurrency and CPU allocation in OS/2. The process in OS/2 provides a level of multitasking which roughly corresponds to a running program. A process consists of one or more threads. This means that in addition to having multiple programs execute at the same time, a single program may have multiple threads that are executing concurrently. OS/2 controls multitasking by using a preemptive priority-based scheduler. OS/2 also provides a rich set of interprocess communications services which are essential in a multitasking environment. Finally, OS/2 was designed from the ground-up to provide intertask protection, another essential requirement in a multitasking environment.

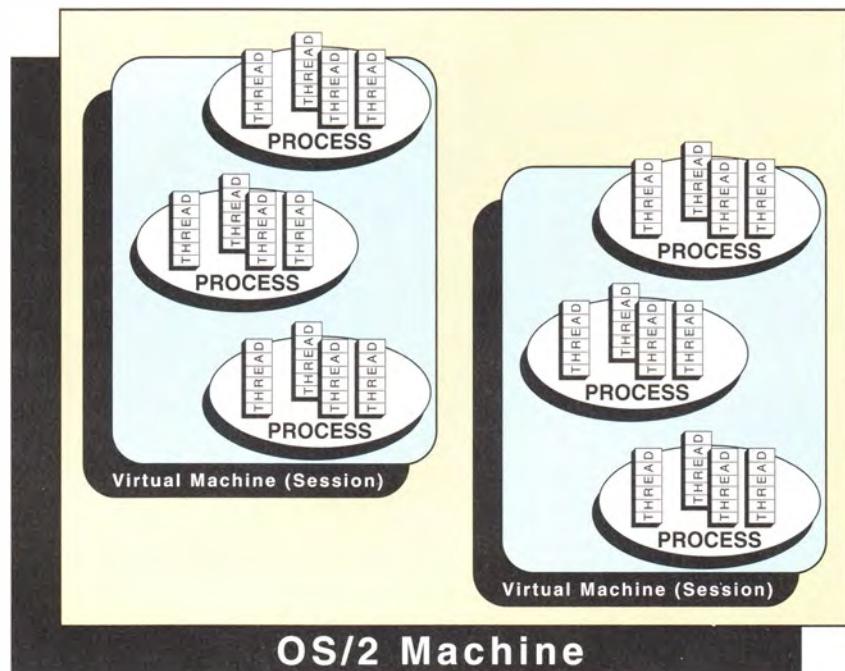


Figure 2. OS/2's Multitasking Hierarchy



REQUIREMENT FROM AN OS	NON-GUI CLIENT		SIMPLE GUI CLIENT	ADVANCED GUI CLIENT
	Without Multitasking	With Multitasking		
Request/reply mechanism (preferably with local/remote transparency)	Yes	Yes	Yes	Yes
File transfer mechanism to move pictures, text, database snapshots	Yes	Yes	Yes	Yes
Preemptive multitasking	No	Yes	Desirable	Yes
Task Priorities	No	Yes	Desirable	Yes
Interprocess communications	No	Yes	Desirable	Yes
Threads for background communications with server	No	Yes	Yes (unless you like the hourglass icon)	Yes
OS robustness including intertask protection and reentrant OS calls	No	Yes	Yes	Yes
Window GUI with menus, scrollbars, etc. (CUA '89 vintage)	No	No	Yes	Yes
Advanced GUI with parallel interactive dialogs, drag-and-drop, multimedia support, and large numbers of window handles (Workplace vintage).	No	No	No	Yes

Table 2. What does a Client Need from an Operating System?

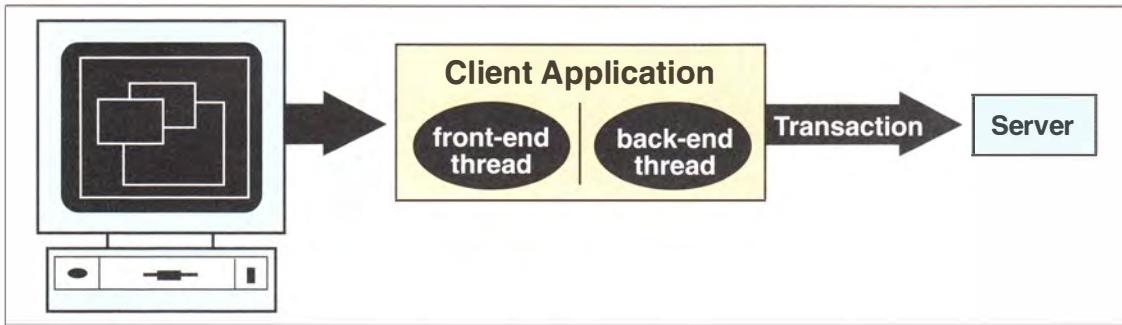


Figure 3. The Anatomy of a GUI Client Application

OS/2's strong multitasking features and its robust operating system environment have made it the platform of choice for database and file server vendors. This server platform will only get better with OS/2 2.0 (the 32-bit operating system platform) which opens up the capability to directly address 4 gigabytes of memory using the flat memory model. This may satisfy (for a few years at least) the ravenous appetites for memory server programs are infamous for.

THE ANATOMY OF A CLIENT PROGRAM

All client applications have something in common: they request the services of a server. What makes client applications different is what triggers the requests and what graphical user interface (GUI, pronounced gooey), if any, is needed. Based on these differences, we can classify clients into three categories: Non-GUI Clients, Simple GUI Clients, and Advanced GUI Clients.

Non-GUI Clients

Non-GUI client applications generate server requests with a minimal amount of human interaction. Non-GUI clients fall into two sub-categories:

- Non-GUI clients which do not need multitasking. Examples are: automatic teller machines (ATMs), bar code readers, and touch-tone telephones. These clients may include a simple human interface in the request generation loop.

- Non-GUI clients which need multitasking. Examples are: robots, testers, and daemon programs. These clients often require very granular, real-time, event driven multitasking services.

Simple GUI Clients

Simple GUI Clients are applications where occasional requests to the server result from a human interacting with a GUI. The simple GUI interface is a good fit for mainstream, OLTP-type, business applications with repetitive tasks and high volumes. They also make good front-end clients to database servers. Simple GUI client applications are graphical renditions of the dialogs that previously ran on dumb terminals. GUIs replace the green screen uglies with graphic dialogs, color, menu bars, scroll boxes, and pull-down and pop-up windows. These applications usually fall under the current Common User Access (CUA) paradigm, also known as the CUA '89 graphical interface². Simple GUI dialogs use the object/action model where users can select objects and then select the actions to be performed on the chosen objects. Most dialogs are serial in nature.

Advanced GUI Clients

Advanced GUI Clients are applications which generate many concurrent requests to the server. These applications are used by information workers doing multiple, variable tasks whose sequence cannot be predicted. Examples include: executive and decision-support applications, multimedia based training systems, operations consoles, and stockbroker workstations. These applications use a workplace metaphor which provides a

Do DOS and Windows support these new demands, or do we need to move on to OS/2?



visual space (think of it as a container) where related objects and programs can be brought together to perform a task. The desktop can contain multiple workplaces running concurrently. Each workplace may be running parallel dialogs (also called modeless dialogs) over parallel sessions with the server.

With advanced multimedia applications these parallel dialogs may be used to display images, video, multi-object folders, and voice annotated mail. Information is displayed to the user in the foreground windows, while background tasks are constantly moving information to and from servers. For example, the first page from a multimedia document is displayed in a window while a background task is busy prefetching the rest of the document from the server. The *workplace* must provide support for:

- The direct manipulation of iconic objects through mouse *drag-and-drop* techniques.
- Advanced interobject communications.
- Parallel dialogs over concurrent information channels.

WHAT DOES A CLIENT NEED FROM AN OPERATING SYSTEM?

Each of the three different types of clients described above place a different set of requirements on the operating system. These requirements are listed in Table 2. As you can see, all client applications need some mechanism to communicate requests and files to a server. Most client categories will function best in a robust, multitasking environment.

CLIENT TYPE	PREFERRED PLATFORM	WHY?
Non-GUI Client (non-multitasking)	DOS	DOS is the low-cost solution.
Non-GUI Client (multitasking)	OS/2	OS/2 can handle real-time event driven, preemptive multitasking requirements. And, it is more robust.
Simple GUI Clients	OS/2	OS/2 provides threads to handle the server requests whereas Windows would show too many hourglass icons in client-server environments. PM and Windows are both adequate GUIs. OS/2 is more robust. Windows allows too many Unrecoverable Application Errors (UAEs) requiring a system reboot.
Advanced GUI Clients	OS/2	OS/2 provides threads to handle the server requests, preemptive multitasking, semaphores, and advanced interprocess communications all of which are needed to support parallel dialogs. PM provides some API support for drag-and-drop. OS/2 is more robust. UAEs are not acceptable.

Table 3. Client Platform Choice: Comparing DOS-Windows and OS/2-PM



It is particularly important for the client environment to be robust because it is impossible for client-server vendors to test client software on all possible hardware/software combinations. It is important to use an operating system that can protect programs from clashing and crashing. No client program should cause the system to hang (requiring a reboot).

GUI clients work best with a thread-like mechanism for handling the background requests (see Figure 3). By using separate threads for the user interface and background processing, the program can respond to user input while a separate thread handles the interaction with the server. This is how GUIs avoid the notorious hourglass icon, a sure sign that the computing environment is not keeping up with the human. Priority-based preemptive multitasking is also required to respond to multimedia devices and to create client applications where multiple dialogs are displayed in parallel.

THE CLIENT PLATFORM: OS/2 OR WINDOWS?

For most software developers, the choice of a client will be determined by the installed base, not the requirements of the client application. Based on this criterion today's client platform of choice will most frequently be DOS-Windows. However, if we were to focus on the technical requirements discussed above, a different picture emerges. Table 3 shows that OS/2-PM is the client platform of choice for three out of four categories of client applications.

CONCLUSION

The title of this article asks, "Client-Server Solutions: OS/2 or Windows 3.0?" The answer to this rather simplistic question would have to be OS/2. It comes closest to providing all the necessary platform ingredients needed to create mission-critical client-server solutions. But, as we have shown, client-server applications are not all alike. So, we need to fine-tune this answer.

Let's first answer, "Servers: OS/2 or Windows 3.0?" Clearly, OS/2 is the preferred server platform, regardless of which operating system the clients are using. OS/2-based servers can replace both the overworked DOS server machines and minicomputers at PC prices. Anyone putting together a client-server application on a PC LAN will almost have to run OS/2 on the server machine.

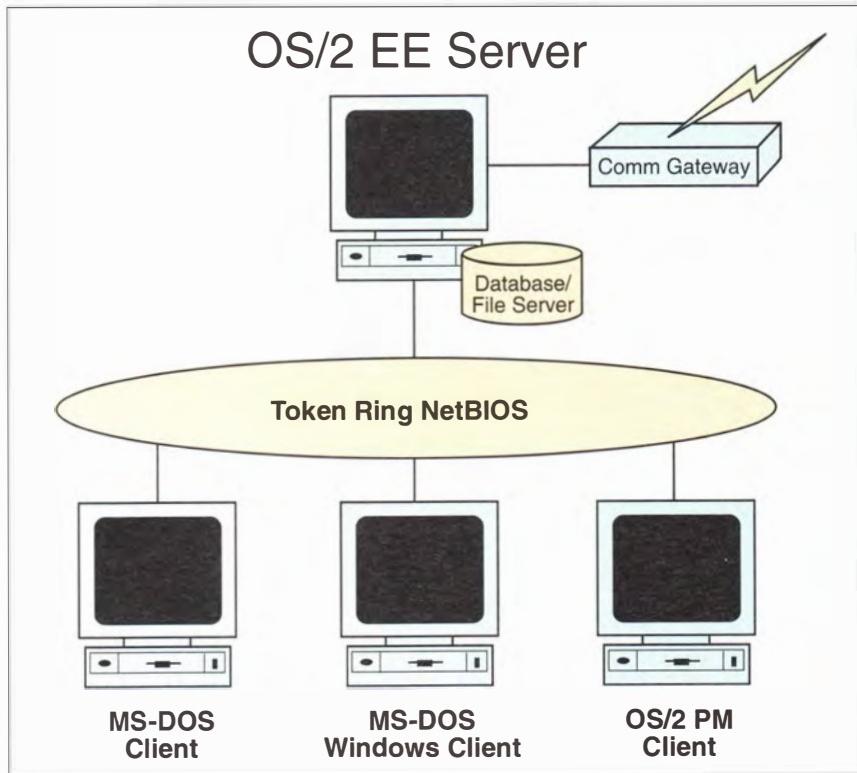


Figure 4. OS/2 EE as a Super Server for Mixed Clients





Does the choice of OS/2 on the server machine dictate a choice of OS/2 on the client machine? Fortunately, the client-server platform allows us to mix and match clients and servers almost seamlessly. And, OS/2 EE servers blend well with DOS and Windows clients. Figure 4 shows a mixed operating system environment.

The next question we need to answer is, "Clients: OS/2 or Windows 3.0?" If your client is GUI-based or requires multitasking then OS/2 provides a better platform (provided it supports the other applications you may be interested in such as spreadsheets and word processors). OS/2-PM clients are more robust, more maintainable, more responsive, and more capable of handling the requirements of complex multimedia environments than DOS or Windows clients. A PM application designer can schedule computational or I/O bound activities as separate threads to ensure crisp user interaction at all times. These are all very desirable features in complex networked environments. For simple non-GUI clients that do not require multitasking DOS is the preferred client platform.

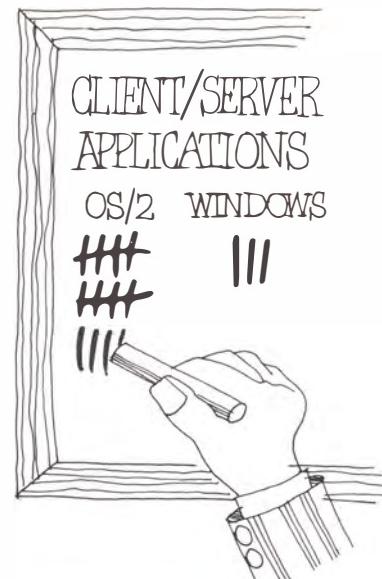
So, the answer is OS/2 on both clients and servers for all but the most trivial client-server applications. OS/2 satisfies current needs and provides a migration path for future information-intensive applications. As the art of creating client-server applications becomes better understood, we anticipate that the bulk of today's client applications on DOS and Windows 3.0 will gradually be replaced with OS/2-PM.

REFERENCES

1. Orfali, Robert, and Dan Harkey. Client-Server Programming With OS/2 Extended Edition. New York: Van Nostrand Reinhold, 1991. (This book may be ordered from IBM Mechanicsburg as publication number G325-0650).
2. SAA Common User Access Advanced Guide. IBM Document SC26-4582, June, 1989.

Robert Orfali, IBM Corporation, 5600 Cottle Road, San Jose, CA 95193. Mr. Orfali is an advisory programmer in IBM System Storage Products Division, Advanced Systems Development. He joined IBM in 1972 and is the architect of TxE, a client-server "Transaction Enabler" platform. Mr. Orfali is co-author of *Client-Server Programming with OS/2 Extended Edition* and received an MSEE degree from the University of California at Berkeley.

Dan Harkey, IBM Corporation, 5600 Cottle Road, San Jose, CA 95193. Mr. Harkey is an advisory programmer in IBM System Storage Products Division, Advanced Systems Development. He joined IBM in 1977 and is the lead developer of TxE, a client-server "Transaction Enabler" platform. Mr. Harkey is co-author of *Client-Server Programming with OS/2 Extended Edition* and received an MSCS degree from Santa Clara University.



OS/2 LAN Support

LAN Application Certification Support

by Art Borrego and Michael Roth

"As a developer my goal is to increase the install base of my application. I also want my product to be well accepted by its users in the OS/2 LAN environment."

If you are an application developer and this statement is true for you, then this article will be very beneficial. It explains how the new LAN Application Certification Support program works. Through this support you may be able to obtain more exposure for your application through IBM's National Solution Center application solution database accessed by marketing representatives. The acceptance of your application may also result in increased quality, and better installation and configuration documentation.

AN Application Certification Support is a new program offered by the IBM Application Development Support Group in Austin, Texas. This support is offered free of charge to Developer Assistance Program (DAP) members who have applications they want to certify in the OS/2 LAN Server environment. Through this support program, vendors certifying their applications receive *IBM OS/2 LAN Application Certification Guidelines*, direct technical support, and sometimes access to our OS/2 LAN test lab facilities in Austin.

Many DAP members have received a mailing about this program. The mailer includes *IBM OS/2 LAN Application Certification Guidelines* along with the National Solution Center (NSC) Application Nomination form. If you do not receive this mailer and are interested in obtaining more information, please submit a request for more information through IBMLINK. To do this, select option 11, LAN Software, from the Topic Selection menu.

Be sure to use the key words "LAN APPLICATION CERTIFICATION SUPPORT" in the brief description section of the 'Ask A Support Question' panel.

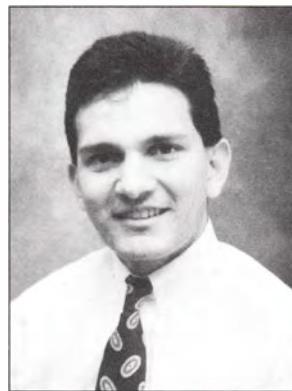
IBM OS/2 LAN Application Certification Guidelines discusses considerations and procedures that should be focused on when certifying an application in the OS/2 LAN Server environment. Considerations discussed include API calls made by the application, file accessing, file locking, data integrity, and functional anomalies. The book also discusses installation, configuration, testing, and documentation procedures. The list of considerations along with the procedures outlined in the guidelines should be used to develop a test plan specific to the application.

The certification guideline procedures are divided into three sections: Pre-Installation, Application Installation, and Application Operation.

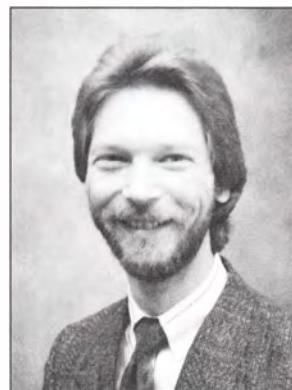
PRE-INSTALLATION

Pre-Installation covers areas that should be considered before the application installation or other testing begins. In this section the first thing to verify is that all the API calls are supported. *IBM OS/2 LAN Application Certification Guidelines* provides references to the IBM publications which contain the lists of supported API calls.

Pre-Installation also includes installing, referencing, and configuring any hardware required by the application. The operating system along with any required components should also be installed and configured on the LAN Server and requesters.



Art Borrego



Michael Roth



APPLICATION INSTALLATION

Application Installation covers the application installation, accessing profile definitions, and administration steps. The application installation procedure is broken down into three types of application installation. Table 1 lists the combinations of installing at a LAN requester to a local drive. Other types of application installation are: installing at a LAN requester to a redirected drive (local drive on the LAN Server), and installing at the LAN Server to a local drive.

APPLICATION OPERATION

Application Operation covers the suggested operational testing for an application in the OS/2 LAN environment. Here certification considerations focus on verifying functionality, file accessing, file locking, and data integrity. Application Operation is divided into three sections: Single User, Multiuser, and Load/Stress.

Install at a LAN requester to a local drive	DOS Application	OS/2 Application	Family Application
DOS requester	●		●
OS/2 req in DOS Mode	●		●
OS/2 req in Protect Mode		●	●

Table 1. Combinations of Installing at a LAN Requester to a Local Drive

SINGLE USER OPERATION

Single user operation testing verifies basic application functionality. You should develop a test case that achieves thorough operational coverage of the application. In this testing the goal is to verify that all application features function as expected and all possible API calls are made. It is advisable to check that sufficient work space is available, particularly with DOS applications running either on DOS requesters or in DOS compatibility mode of OS/2 requesters. Local and remote printer/serial device access should also be tested if used by the application.

MULTIUSER OPERATION

MultiUser operation testing verifies file accessing, file locking, and printer/serial device sharing. A minimum of three requesters should run the application concurrently and access common files and devices from an OS/2 server.

LOAD/STRESS

While going through the Load/Stress testing, verify that the application functions correctly under load and stress conditions. This testing is similar to multiuser testing but includes heavy access of the application executables and data files from the same OS/2 server. As suggested in the guidelines, all printing and serial device sharing should also be redirected to the same server. A minimum of six LAN requesters, varying in hardware configuration, and continuously running the application tests for an extended period of time (8 hours or more), are suggested by the guidelines.

DOCUMENTATION

Documentation is key while going through the certification process. Appendix A of *IBM OS/2 LAN Application Certification Guidelines*, describes the Certification Activity Documentation worksheet format and contains a sample worksheet. While going through the certification process, you should document all pertinent certification activity information. This information includes: hardware and software requirements, installation, configuration, and setup steps required for the application in the OS/2 LAN environment.

The Certification Activity Documentation worksheet for your application should be submitted along with the NSC Application Nomination form. If approved, the information supplied in the NSC form, along with the worksheet, will be entered into the NSC application solutions database.

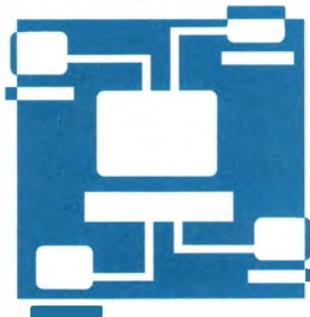
The NSC database is an important tool for IBM sales and support personnel when searching for an application to meet their customer's needs. For example, a customer is installing an OS/2 LAN at their site and also wants a LAN facsimile application. The IBM marketing representative would submit a query with the key words "OS/2, LAN, facsimile" against the database. This action would retrieve a list of certified OS/2 LAN facsimile applications.

This is an excellent channel to expose DOS or OS/2 applications compatible in the OS/2 LAN Server environment to IBM sales and support personnel. Take advantage of this program today!

Art Borrego, IBM Personal Systems Line of Business, 11400 Burnet Road, Austin Texas 78758. Mr. Borrego is a senior associate programmer working in OS/2 EE and LAN Server Application Development Support providing direct technical support to software vendors writing OS/2 EE application, in the LAN environment. He is a coauthor of *IBM OS/2 LAN Application Certification Guidelines*, and coordinator of the OS/2 LAN support facilities in Austin, TX. He joined IBM in September 1987 and worked in the multi-environment testing of OS/2 EE for three years. He has a BS in Computer Science from the University of Texas at El Paso.

Michael Roth, CDI Corporation, 11000 Metric Blvd, Austin, Texas 78758. Mr. Roth has 19 years experience in the defense and computer industries, including communications operating systems development, LAN/communication systems, and system/performance tests. He has provided technical services to IBM in a variety of OS/2 development activities and is coauthor the *IBM OS/2 LAN Application Certification Guidelines*. Mr. Roth studied Electrical Engineering and Computer Science at the Massachusetts Institute of Technology.





OS/2 Multi-User

Remote-OS Multi-User System for OS/2



Walter Kac

by Walter Kac and Alan King

This article is part of a series on multiuser extensions for OS/2™. Previous issues of the Developer included articles on the IBM RPG II Application Platform® and Polymod2®.

Remote-OS®, from The Software Lifeline Inc.® (TSLI) extends the reach of applications to disparate types of terminals and their access to many types of systems on the network. It permits up to 32 users to share a single PS/2®, and offers easy migration to the world of OS/2.

TSLI's software enables APPLE®, DOS PCs, LAN stations, ASCII terminals, UNIX® or AIX® PCs and 3270-type terminals to access most text-based applications on the OS/2 Host PC, whether locally or remote-attached. Remote-OS supports Windows3® PCs as terminals, thereby enabling remote or local access to the OS/2 CPU.

Remote-View™, which ships with Remote-OS, allows any terminal to view the foreground and background sessions of any terminal attached to the Host PC. This allows one terminal to view up to eight terminals concurrently or, up to 32 terminals to view eight terminals concurrently.

Using Remote-OS, terminal-attached printers and file transfers to the DOS station bring home almost all of the features of LANs. Terminal protocols are defined from the system console or an active terminal (PC or ASCII Non-Intelligent Display) through the OS/2 System Editor. A set of pre-defined workstations is provided within the software, including VT220. Additional ASCII terminals may be defined via the tool kit. The standard COM ports are supported, as well as multi-port cards like OCTOPORT®, HOSTESS® and AST.

Remote-OS is totally transparent to the user, therefore there are no operational differences when using the "console" display or the remote terminal itself. It even offers the ability to set up a menu shell in the first screen group, thereby enabling the user to hot-key to a menu and start or return to applications. The <Ctrl Esc> and <Alt Esc> functions of OS/2 are also available to the terminal, enabling each terminal user to run up to eight applications.

Only basic knowledge of OS/2 and its editor are necessary to set up the Remote-OS system.

Remote-OS can be used as a gateway to Novell®, 3COM®, IBM LANs® and many others. When Remote-OS is also a gateway, a remote terminal or PC can access the LAN environment as if that terminal was actually a member of the LAN, mainframe or VAX® system on DECNet®. Remote-OS provides remote access to files, spooled printers, program execution, interprocess communications, distributed data bases, etc.

APPLICATION SUPPORT

Remote-OS can be used with any application in multi-user mode, as long as the application programs are:

- Native to OS/2;
- Written in text mode so that they display the Standard PC semi-graphic character set;
- Able to start several times simultaneously, as required by OS/2; and
- Written according to the standards published by IBM and Microsoft.



Remote-OS comes with an encryption device that can be programmed for application or user access protection, making it virtually impossible to "break into" the network.

INTEROPERABILITY

Figure 1 exemplifies the interoperability achieved when Remote-OS is the application enabler. All levels of terminals, (like DOS PCs, ASCII or 3270 terminals, Apple Macintosh®, and so forth,) can access any system and/or application within or outside of the Enterprise. With this seamless architecture, there is only one platform to support. End-users need minimal education and are already familiar with the systems and a large number of already-installed applications. Remote-OS offers the ability to protect inventory of installed applications and a natural path to strategic applications is formed, while requiring minimal investment to access this technology. Additionally, Remote-OS will also allow OS/2 "PASS-THRU" to UNIX® Systems.

The terminals supported by Remote-OS can all have PC-style keyboard layouts making it easy for users to switch between using the computer directly and using a terminal. Equally important, application programs designed to use key combinations from a PC-

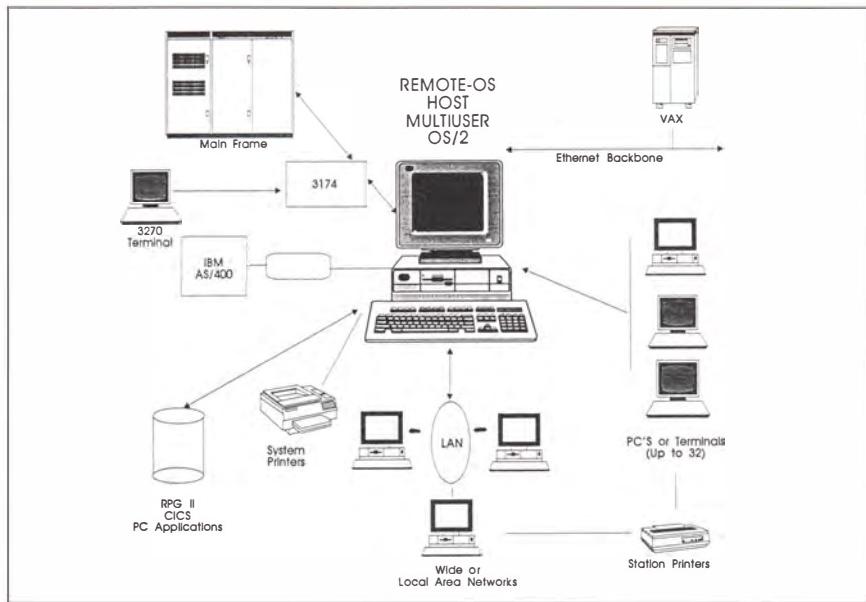


Figure 1. Interoperability with Remote-OS

REMOTE-OS OVERVIEW

OS/2 is essentially a single-user operating system with the capability of being networked with other OS/2 or MS-DOS computers to share databases, files, printers etc. However, each user must have their own fully functional computer.

Remote-OS is a software subsystem which runs alongside OS/2 and allows multiple users to share a single computer and its resources. Each user accesses the central OS/2 computer through the use of a currently installed PC or a low-cost dumb terminal, thus providing significant cost benefits over the networked multiuser solution.

Terminals can be connected into the computer's RS232 serial ports, COM1 through COM8, or into other third-party multi-port adapters. Terminals may be connected directly to the computer or may be cited at remote locations by using modem or packet-switched links.

style computer keyboard function correctly when run at a terminal. However, recognizing the large inventory of non-PC-ready terminals, Remote-OS allows customization of keyboards to accommodate the PC Function keys.

Only programs which run in OS/2 protect mode can be accessed from the terminals through Remote-OS. Thus, DOS programs running in the OS/2 DOS Environment are not available. This will change when OS/2 2.0 is generally available, and DOS will also be available at the terminals.

The terminal programs must also be text-mode, or kernel, programs. These programs appear similar in their screen operation to many DOS programs in that their output is generally alphanumeric text with a limited graphics capability obtained by using the line drawing characters in the extended screen character set. OS/2 protect mode applications designed to run only under OS/2 Presentation Manager are not able to be accessed from a terminal.



Terminal programs may otherwise use most of the facilities available to any other OS/2 program. This includes use of the API (application programming interface) calls for the base operating system, Database Manager™, Communications Manager™, LAN Manager® and many more.

API CALLS, KBD API CALLS, VIO

Terminal input and output is achieved through the use of the standard OS/2 KBD and VIO API calls. Because of this, programs do not have to be specifically designed to run from terminals and do not need to use any non-OS/2 API calls to run happily from a terminal. Indeed, many off-the-shelf applications, utilities and language products already run from terminals under Remote-OS.

If you are running off-the-shelf software under Remote-OS, you must ensure that the license agreement for that software does not impose any restrictions in terms of number of concurrent users. If the license agreement appears to restrict multi-user operation, you should contact the software producer or distributor to clarify, modify or secure additional program licenses.

TERMINAL, PROGRAMMING CONSIDERATIONS

Some off-the-shelf programs, although functionally operational under Remote-OS, may prove slow for terminal usage if they frequently display massive amounts of data or if they redisplay large portions of the screen unnecessarily. On a single-user computer, these screen updates would probably not be noticed because the screen updating is performed directly. However, all data going to a terminal screen has to travel down a serial RS232 line which normally operates at 9600 bits per second, (approximately 1,000 characters per second) or at 19200 bits per second (approximately 2,000 characters).

LOCKING, FILE AND RECORD

Since all well-written OS/2 protect-mode programs are inherently designed with OS/2 multitasking in mind, they should present

no problem under Remote-OS. This is because, even on a standalone and single-user OS/2 system, multiple programs or even multiple instances of the same program may be started by the operator. Each program must therefore protect any file access against it, not just other programs, but multiple instances of itself. Even though all OS/2 programs must provide for file and record locking, some programs might take the easy route and lock an entire file instead of just the section being updated. For a word-processing program, locking an entire document file is probably the correct thing to do, however, for an accounts receivable program, to lock one of the ledger files must be considered a fault on a multitasking system such as OS/2.

MAXIMUM NUMBER OF TERMINALS

Remote-OS supports the attachment of up to 32 terminals to a single host PC. Each version of Remote-OS is licensed for a maximum number of concurrent users which may be less than the number of terminals attached to the computer. If more than the licensed number of users attempt to access the system, a system busy message is displayed at each of the additional terminals and they are put into a waiting queue until one of the current users logs off the system.

The number of concurrent users is controlled by the TermPak hardware module supplied with Remote-OS. The number of concurrent users can be upgraded at any time.

Remote-OS runs under OS/2 Standard Edition or Extended Edition, IBM or Microsoft versions 1.1 through 1.3 and Version 2.0.

Remote-OS will run on Micro Channel Architecture (MCA) bus systems such as the IBM Personal System/2 range, or on IBM PC AT or compatible systems using derivations of the AT bus; these systems are sometimes called Industry Standard Architecture (ISA) bus systems. The processor may be the Intel 80286, 80386SX, 80386 or 80486.

Remote-OS is able to use monochrome or color display adapters such as the MDA, CGA, MCGA, EGA and VGA.

Memory requirements depend very much on the mix of application programs to be run from the computer and the terminals.

However, systems managers should consider using a minimum of 4 megabytes of RAM with the Standard Edition of OS/2 1.1 or 1.2. Remote-OS itself is not particularly memory-hungry, but the code and data requirements for the required mix of programs can add up. If insufficient memory is available, and swapping in the OS/2 CONFIG.SYS file is enabled, the system will remain functional, but response times may deteriorate at the computer keyboard and at the terminals.

Remote-OS—and OS/2 itself—require the system to have a hard disk. Disk capacity depends on the applications installed and the data file and database storage requirements. Remote-OS itself uses very little disk space. The computer must have at least one diskette drive in order for the Remote-OS software to be installed. The diskette drive may be a 5.25 inch 1.2Mb drive or it may be a 3.5 inch 1.44Mb drive.

The computer must have at least one RS232 serial port for connecting a terminal. Other RS232 ports may be added from the following range:

- COM1 through COM8 compatible adapters
- HOSTESS range of 4 , 8 and 16 port adapters
- OCTOPORT range 8 with 2 @ 8 port adapters for 16

At least one terminal is required from the following selection :

- IBM 3151 ASCII Display Station models 31, 41, 51 or 61 (the models 31 & 41 must use the Cartridge For Connectivity)
- WYSE® WY-60 or WY150
- WYSE WY-50 or WY30 terminals that have a compatible PC-Terminal mode (adapted using the tool kit)
- VT220 with the VT220 ATD file supplied with Remote-OS
- IBM 3270 Terminals attached via the 3174 equipped with AEA



INSTALLING REMOTE-OS

The installation procedure is mostly automated and requires the operator to start just one program. The communication adapters to be used by Remote-OS should already have been installed in the computer and configured.

The Remote-OS package (5 and above user versions) contain a hardware module called the TermPak which controls the number of users able to concurrently access the computer. The TermPak must be inserted into one of the RS232 serial ports controlled by Remote-OS. Initialization and configuration are controlled by statements contained in a file named TRMMAN.INI. The port to which the TermPak is connected may be reserved solely for the TermPak or may also have a terminal connected to it; the TermPak module does not affect the operation of the terminal in any way. Figure 2 illustrates a sample init file.

```
::::::::::::::::::;;
; System Section
::::::::::::::::::;;
[SYSTEM]
SHELL = C:\C:\OS2\CMD.EXE
..KEYBOARDDCP = C:\C:\OS2\KEYBOARD.DCP
TERMPAKPORTNO = 1
;
; Ensure the drive/directory are correct
; Ensure the drive/directory are correct
; Ensure the TermPak is attached to the
; .. logical port no 1 (COM1 in this case)
; IGNORE THIS STATEMENT FOR REMOTE-OS2
; TWO-USER VERSION.
;
::::::::::::::::::;
```

Figure 2. SAMPLE.INI (continued)



```

; PORT Section
::::::::::::::::::;;
[PORT]
    ADAPTER=COM1 ; Names the type of adapter
    PORTNO=1 ; Assigns logical port no. 1 to the port
    IRQ=4

CMore:
;TERMINAL Section
::::::::::::::::::;;
[TERMINAL]
    TERMNO      = 1 ; Assigns a logical terminal number
    PORTNO      = 1 ; Names the logical port it's attached to
    LOCATION    = 'Attached to COM1' ; Change for desired description
    TERMTYPE    = DOS-TERMINAL ; Change for terminal type
    KBDTYPE     = 1,US ; IBM3151, WYSE60, WYSE150
    FORCETS     = YES ; DOS-TERMinal, ASCII-VT220, 3270
    FORCEDSR    = YES ; 0 - Keyboard with F1-F10 keys
    FORCEDCD    = YES ; 1 - Keyboard with F1-F12 keys
    BAUD=9600; ; Ensures no initial cable problems
    MAXSESSIONS=8; ; ... however these should be changed
    XONXOFF=YES; ; ... when you are sure the cables are
    ; ... correctly wired
    ; SHELL=C:\TRMMAN\TRMShell.EXE; ; USED FOR STARTING A UNIQUE BATCH
                                    ; OR .CMD FILE FOR EACH TERMINAL.
                                    ; EXAMPLE: START001.CMD WILL EXECUTE
                                    ; FOR TERMINAL #1 WHEN TERMINAL BOOTS

```

Figure 2. SAMPLE.INI

TRMMAN.INI may be modified to reflect local configuration of communication adapters, number of terminals and many other options. After installation, TRMMAN.INI is set up for a configuration of one terminal attached to the COM1 port. It is recommended that you experiment using this configuration before setting up TRMMAN.INI for your own specific configuration. The correct cable configuration is required for flow control, or pacing of data between the computer and the terminal. For ease of experimentation, the installed TRMMAN.INI for COM1 will operate with just a basic cable with *Signal Ground*, *Transmit Data* and *Receive Data* connected. However, this cable is not able to detect a terminal being switched on or off and always assumes that a terminal is present and ready to receive data.

The installation procedure described above is automatic and does not require users to be aware of which files need to be copied or

what changes should be made to CONFIG.SYS. TRMMAN.INI must be located in the directory reserved for Remote-OS, i.e. its full path name must be \TRMMAN\TRMMAN.INI on the boot drive. The file is read by both the device driver TRMMAN.SYS and by the main Remote-OS program TM.EXE.

TRMMAN.INI is structured in the standard format of .INI files. It is divided into a number of sections each starting with a key word enclosed in square brackets '[....]'. Within each section a number of statements are specified, each starting with a key word followed by a parameter value. The file may contain comment lines starting with the ';' character; anything following the ';' character on any line of the file is treated as a comment and is not used by Remote-OS. Blank lines are ignored and may be inserted to aid readability.

SYSTEM SECTION

The SYSTEM section identifies certain system-wide characteristics of the environment in which Remote-OS is running and contains information used by both TRMMAN.SYS and TM.EXE. If the SYSTEM section is omitted, default values are assumed. For example:

BUS= specifies the type of bus architecture used by the computer.

SHELL= specifies the default shell program activated whenever a terminal comes on-line.

KEYBOARDCP= specifies the default code page file used for translating keystrokes to internal character values.

TERMPAKPORTNO= specifies the adapter port to which the TermPak module is connected.

POR T SECTION

The PORT sections identify the characteristics of the physical communication adapters to be used for Remote-OS and contain information read by TRMMAN.SYS at computer boot time. The PORT sections allow logical port numbers to be assigned to each physical port on the adapters. The PORT sections are used by TRMMAN.SYS for allocating memory for transmit and receive buffers and for initializing associated variables. The TRMMAN.INI file contains as many PORT sections as there are adapter ports to which terminals may be connected. Changes made to PORT sections do not become effective until the computer is rebooted. For example:

PORTNO= assigns a logical port number to an adapter port

ADAPTER= specifies the type of communication adapter

IRQ= specifies the interrupt request level used by the adapter

IOBASE= specifies the I/O port address used by the software to talk to the adapter

TXBSZ= set the size of the transmit buffer for the port

TXBLO= specify the low trigger point at which processes waiting for space in the buffer will be run

RXBSZ= set the size of the receive buffer

RXBHI= specify the trigger point at which the flow control procedure will stop reception of data i.e. when to send XOFF and/or drop the DTR signal

RXBLO= specify the trigger point at which the flow control procedure will restart reception of data i.e. when to send XON and/or raise the DTR signal

TXFIFO= optimize the operation of the first-in first-out transmit buffer used on some communication adapters

RXFIFO= optimize the operation of the first-in first-out receive buffer used on some communication adapters.

BUS, AT // BUS, MCA

With the exception of the HOSTESS and OCTOPORT range of adapters, sharing of interrupts on an IRQ level between physically separate boards is not supported on PC AT compatible computers. Computers based on the AT bus may have individual ports sharing a single IRQ level only if all the ports are physically on the same board i.e. if the board is designed to support IRQ sharing within the bounds of the board itself. The HOSTESS and TCL adapters for the AT bus provide for connections between separate boards to implement IRQ sharing. Adapters for the PS/2, and compatible computers using the MCA bus, may share IRQ levels.





TERMINAL SECTION

The *terminal* sections contain information used by TM.EXE to identify and configure itself for each attached terminal. Each terminal section assigns a logical terminal number to a terminal and associates it with one of the logical port numbers assigned in a PORT section. There will normally be the same number of TERMINAL sections as there are PORT sections although, if a terminal is not attached to one of the ports, no TERMINAL section will be present for the PORT. For example:

TERMNO= assigns a logical terminal number to the terminal

PORTNO= specifies to which port the terminal is attached

LOCATION= specify a description that may be used to indicate the location or user of the terminal

TERMTYPE= specifies the type of terminal

KBDTYPE= specifies the type of keyboard attached to the terminal

STATUSLINE= allows you to select whether the status line on the terminal will be displayed or hidden

DEFAULTROWS= specifies the default number of rows, or lines, for the terminal screen

DEFAULTCOLS= specifies the default number of columns for the terminal screen

MAXROWS= specifies the maximum number of rows, or lines, for the terminal screen

MAXCOLS= specifies the maximum number of columns for the terminal screen

SHELL= names the program to be run whenever the terminal first comes on-line

MAXSESSIONS= allows you to set the maximum number of sessions able to be active at any one time on the terminal

BAUD= specifies the baud rate for which the terminal has been configured

DATABITS= specifies the number of data bits in each character exchanged with the terminal

PARITY= specifies the type of parity checking being used by the terminal

STOPBITS= specifies the number of stop bits in each character exchanged with the terminal

XONXOFF= specifies whether the XON/XOFF flow control protocol is to be used

FORCECTS= allows the RS232 Clear To Send signal to be considered permanently high

FORCEDSR= allows the RS232 Data Set Ready signal to be considered permanently high

FORCEDCD= allows the RS232 Data Carrier Detect signal to be considered permanently high

TOGGLERTS= specifies whether the RS232 Request To Send signal is held permanently high or "toggled" on and off

TOGGLEDTR= specifies whether the RS232 Data Terminal Ready signal is held permanently high or is toggled on and off

IDLETIMEOUT= sets the length of the idle period after which the terminal will be logged off

ACTIVATE= specifies whether the terminal is to be activated immediately when Remote-OS is first started



CONFIGURATION

Some terminal types can be configured for screen sizes other than the standard 80 columns by 25 rows (e.g. 132 columns by 25 rows, 80 columns by 43 rows etc.). The *defaultcols* statement specifies the number of columns that will be initially set for each new session starting on the terminal. Once a session is started, an application program running in that session may issue the VioSetMode API call to change the number of columns and/or rows.

Additionally, where ASCII keyboards generate pure ASCII codes instead of scan codes, Remote-OS must treat these ASCII codes in a special way so that they may be presented to an application program as though they had come from a genuine PC keyboard. This special handling must also include a means of simulating the pressing of keys that are present on PC keyboards but which may not be present on ASCII keyboards. Remote-OS therefore performs the task of converting combinations of ASCII keystrokes into a sequence of keystrokes resembling those generated by a PC keyboard.

In general, a PC keyboard is able to generate many more sequences than a pure ASCII keyboard and it is likely that a number of keys present on a PC keyboard may not be able to be represented or simulated on an ASCII keyboard. The Programmer's Toolkit (included with Remote-OS 5-33 user versions) includes a utility to generate a file that controls Remote-OS's processing of ASCII keystrokes and also defines the screen control codes. This file is termed the ASCII Terminal Definition file and normally has the extension .ATD. See Figure 3.

The shell program is the name given to the first program loaded whenever a terminal comes online. Normally, the shell program will prompt the operator for a user name and password. Remote-OS allows the system

operator to specify a different shell program for each terminal if required. The *shell* statement in the terminal section specifies the name of the shell program for the terminal being described. Different shell programs may be specified in other terminal sections. The full path name of the program must be given. If the *shell* statement is omitted in the terminal section, the default shell program specified by the *shell* statement in the SYSTEM section is assumed.

The following is an example of the use of the *shell* statement :

```
SHELL = C:\OS2\CMD.EXE
```

The *maxsessions* statement specifies the maximum number of sessions that the terminal is able to support at any one time. The value may be between 1 and 8.

The *baud* statement informs Remote-OS of the baud rate used by the terminal. The parameter following the BAUD= keyword may be one of the following: 56000, 38400, 19200, 9600, 7200, 4800, 3600, 2400, 2000, 1800, 1200, 600, 300, 150, 134, 110, 75, 50.

REMOTE-OS DESIGN CONSIDERATIONS AND ARCHITECTURAL SUMMARY

Remote-OS has been designed as an open extension to the base OS/2 operating system. As such, it allows software houses, VARs and end-users to configure the system for their own specific requirements.

Remote-OS runs on top of out-of-the-box OS/2 with no modifications or hacking of the base operating system. Both the IBM and Microsoft flavors of OS/2 are supported from version 1.1 through version 1.3, as will be OS/2 2.0 when generally available. Because Remote-OS runs on top of OS/2, none of the functionality of Standard Edition or Extended Edition is forfeited.



```
[KEYDATA]
;
; Define the meta keys
;
8F,'P' = META_SHIFT ;PF1
8F,'Q' = META_ALT ;PF2
8F,'R' = META_CTRL ;PF3
8F,'S' = META_CLEAR ;PF4
;
; Function keys F1 to F12
;
9B,'17~' = 112 ;VT-F6=F1
9B,'18~' = 113 ;VT-F7=F2
9B,'19~' = 114 ;VT-F8=F3
9B,'20~' = 115 ;VT-F9=F4
9B,'21~' = 116 ;VT-F10=F5
9B,'23~' = 117 ;VT-F11=F6
9B,'24~' = 118 ;VT-F12=F7
9B,'25~' = 119 ;VT-F13=F8
9B,'26~' = 120 ;VT-F14=F9
9B,'28~' = 121 ;VT-F15=F10
9B,'29~' = 122 ;VT-F16=F11
9B,'31~' = 123 ;VT-F17=F12
```

Figure 3. A page from ASCII.TXT

An optional shell program is included with Remote-OS that requests the user name and password and acts as the log-on mechanism to invoke the built-in security features of the User Log-on Management feature of Remote-OS. Under User Log-on Management, users may be classified as members of pre-defined groups in much the same way as the User Profile Management feature of the base OS/2. The User Log-on Management facilities may also be accessed via API calls to the Remote-OS system.

DEVICE DRIVERS

Remote-OS includes its own set of device drivers for the various communication adapters supported by the product. These drivers are built around an architecture designed to cater for the exacting requirements of character-based terminal input/output. During the development phase of Remote-OS, the standard COM drivers were investigated for their suitability for use as terminal drivers; these drivers are

known as category 1 (or CAT1) device drivers. The decision went against the CAT1 drivers for two reasons; performance limitations and lack of required functionality.

The CAT1 driver architecture is perfectly adequate for block-based communications concurrently using a small number of ports, however, it imposes an unacceptable overhead for character-based communications typical in multiuser terminal-based systems. Each byte sent to or coming from the terminal may, in the worst case, involve a separate call to the operating system to access the device driver. This operating system call goes via the OS/2 file system and permeates down through levels of operating system code until it finally reaches the device driver which is then able to process the request and return the response back up though the operating system layers. Even using intelligent communication adapters does not overcome this fundamental problem, since the on-board processor is only able to take over once its device driver receives the request from the operating system.

The Remote-OS driver architecture uses a Direct Memory Access technique involving just a memory read or write to reach the device driver code. The architecture was designed to take full advantage of both intelligent and non-intelligent communication adapters.

TERMINAL SUPPORT

Remote-OS supports a wide range of dumb terminals having a PC Mode of operation. PC Mode terminals are ideally suited to multiuser OS/2 operation since they usually have PC-style keyboard layouts and more importantly generate the same make and break scan codes that an application program might be expecting from a standard PC keyboard. In addition to the generic PC Mode terminal support, Remote-OS provides specific support for certain named terminals so that features peculiar to that terminal may be fully utilized. This includes use of LED status lights, additional screen status lines and support for various screen sizes e.g. 80 x 132, 43 x 80, 43 x 132 etc. Applications may use the standard VIOxxx API calls to inquire and set the screen size.

The Remote-OS software includes a DOS program allowing DOS-based PCs to be connected in place of terminals with full support for color screens.

In addition to supporting PC Mode terminals and DOS PCs, it was anticipated that Remote-OS would be used to upgrade existing multi-user installations that may be using non-PC Mode terminals. The ASCII Terminal Support (ATS) facility allows regular ASCII/ANSI terminals such as the DEC VT220 or the WYSE WY-50 to be connected to the host OS/2 system. The user is able to specify the characteristics of these ASCII terminals so that almost all non-PC Mode terminals are able to be defined via a straightforward table. Part of the ATS support provides a novel means of mapping the keys on the ASCII terminal keyboard to appear to the application program as though

coming from a regular PC keyboard. This allows the ASCII terminal keyboard to simulate complex PC keystroke combinations such as <Alt key>, <Ctrl key>, Shift key>, functions keys, extended keys etc.

The terminal support for PC Mode terminals, DOS PCs and regular ASCII terminals is optimized to generate minimal traffic on the RS232 line. If an application displays any portion of the screen while in the foreground session, only the positions that have actually changed are sent to the terminal. Screen updates to background sessions do not generate any RS232 traffic since only the Logical Video Buffer internal to Remote-OS is updated. Many OS/2 programs blindly re-display large areas of the screen for even single byte changes. On standard PCs, the screen is updated at electronic speed and the effect of this unnecessary update goes unnoticed. Without the optimization in Remote-OS, these screen updates would generate considerable traffic at the relatively slow RS232 speeds and would render many applications unusable on terminals.

To further reduce traffic on the RS232 line, repeats of held-down keys are generated at the host rather than on the terminal if the terminal has the capability to turn off repeat keystrokes.



FILE TRANSFER

DOS-PCs connected as terminals are able to exchange files with the host OS/2 system concurrently with screen and keyboard activity at the terminal.

PRINTER SUPPORT

Remote-OS includes a Printer Management System designed to operate alongside the base OS/2 spooler to provide spooling facilities to support terminal-based programs.



Printers connected to the auxiliary ports of the terminals may be defined as pooled devices and may be used to output any queued job having the appropriate attributes. Output to terminal-attached printers is interleaved with any terminal keyboard and screen activity.

Each print job is generated with a series of attributes specifying the characteristics of the printer on which it may be printed in addition to some scheduling attributes specifying when it may be printed. Printer attributes include form type, printer type, printer location, printer ownership; while scheduling attributes may specify a hold until date/time and a print-between time. Print jobs may be prioritized by up to nine levels.

Each session on a terminal has its own default print job attributes which may be set by user commands or from an application program via a series of API calls.

LOW MEMORY OVERHEAD

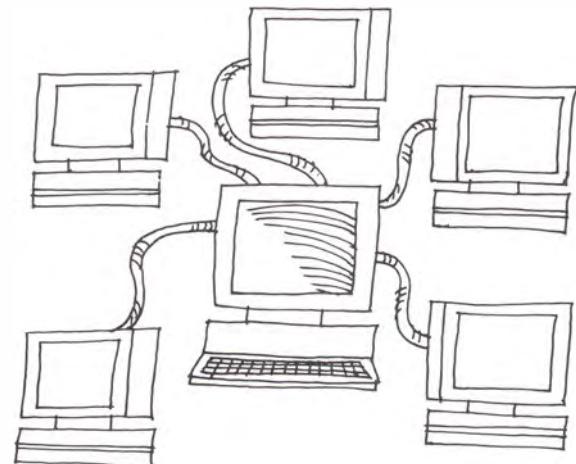
The core of Remote-OS code is written in 80x86 assembler in order to optimize on memory usage and throughput. Again, the design aim of supporting large terminal networks dictated low memory overhead to allow more room for applications, while minimizing processor overhead and servicing the terminals, without stealing processor time from the terminal applications themselves or from the host-based applications.

Code segment usage is less than 80Kb with each terminal requiring something in the region of 30Kb of data segment in order to support two active sessions. An example system configured for 32 terminals each running two sessions would require a total of around 1Mb of memory.

The disk space requirement is less than 512Kb for the current release.

Walter Kac, *The Software Lifeline, Inc., Executive Court One, 2295 Corporate Blvd. NW, Boca Raton FL 33431, is the president of TSLI which he co-founded it in 1988. His previous assignments in the marketing organization of IBM include OEM Marketing, Special Bids and, at IBM European Headquarters, Business Practices for External Channels. Mr. Kac is a graduate of NYU, majoring in Computer Sciences.*

Alan King is the founder and Managing Director of QIIQ Ltd. Mr. King has been involved in operating systems and communications software for over 21 years. He is also the chief architect of the Remote-OS kernel and has worked extensively in most areas of the project.



International

OS/2: The International Scene



by George Reader

I have taken over the International Scene column from Cheryl Bullen who has now returned to the USA. Thank you Cheryl, for your valuable help during your time in the UK.

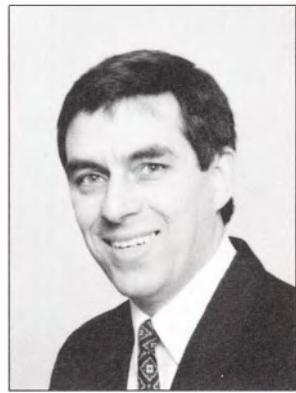
WORLDWIDE DEVELOPER SUPPORT

One of my first tasks has been to update the OS/2 country contact list (see listing on the following pages). As you can appreciate, a list such as this is continually changing. I have tried to make it as complete as possible, including telephone and fax numbers where known. If you have problems reaching anyone, please contact me and I will verify or update the information.

While most IBM world trade organizations don't have a U.S.-style developer assistance program, there is assistance and education available within each IBM country organization to meet varying needs. OS/2 application developers should contact their country representatives with any questions.

OS/2 IN EUROPE/MIDDLE EAST/AFRICA (EMEA)

National language versions of OS/2 Standard Edition 1.3, Extended Edition 1.3 and LAN Server 1.3 became generally available in EMEA countries during the first quarter of 1991. OS/2 Programming Tools and Information Version 1.2/1.3 (English version only) is also available both as a full package and as an update for owners of the 1.2 Toolkit.



George Reader

HOW TO ORDER THE IBM PERSONAL SYSTEMS DEVELOPER

To subscribe to the *IBM Personal Systems Developer*, contact your IBM representative and request that your subscription be entered through your country literature coordinator on the IBM SLSS ordering system. This system resides in IBM internal libraries worldwide. Place a subscription order for G362-0001 and all future issues will be sent to you.

We welcome any comments and suggestions for international topics or article submissions for the Developer magazine



OS/2 COUNTRY CONTACTS

Country	Contact	Details	Location
Asia Pacific Group			
Australia	Mel Steiner	Tele:61-1-634-8991 Fax:61-2-680-4285	Sydney
Japan	W. Kumada	Tele:81-3-3808-4290 Fax:81-3-3664-4951	Tokyo
Americas			
Canada	Tom Allan	Tele:1-416-443-5248 Fax:1-416-443-4746	Toronto
Mexico	Juan Carlos Fernandez Sarda	Tele:52-5-557-8588 x1846 Fax:52-5-395-7812	Mexico
Europe, Middle East and Africa Corporation (EMEA)			
Austria	Hans Dufek	Tele:43-222-21145 x2731 Fax:43-222-2160886	Vienna
Belgium	Patrick Bulckaen	Tele:32-2-214-2386 Fax:32-2-218-6955	Brussels
Denmark	Poul Hansnes	Tele:45-45-93-4545 x4522 Fax:45-45-93-2420	Copenhagen
Egypt	Shamel Abaza	Tele:20-2-349-2533 x723 Fax:20-2-703155	Cairo
Finland	Ilkka Ayravainen	Tele:358-0-4594007 Fax:358-0-4595772	Helsinki

Figure 1. List of OS/2 Countries and Contacts (Continued)



Country	Contact	Details	Location
Europe, Middle East and Africa Corporation (EMEA) (Continued)			
France	Jean-Paul Rambaud	Tele:33-1-49057390	Paris
Germany	Hans-Michael Obst	Tele:49-711-785-2417 Fax:49-711-785-3483	Stuttgart
Greece	Chryssina Hadjitheodorou	Tele:30-1-32-219769 x706	Athens
Gulf Countries	Robert Kikano	Tele:973-210880	Bahrain
Iceland	Helgi Petursson	Tele:354-1-697769 Fax:354-1-690377	Reykjavik
Ireland	Noel O'Rorke	Tele:353-1-603744 x4361 Fax:353-1-600638	Dublin
Israel	Arnan Dar	Tele:972-3-618-700 Fax:972-3-618-211	Tel Aviv
Italy	Virgilio Ferrari	Tele:39-2-7548-2957 Fax:39-2-753-2326	Milan
Netherlands	Frans Bik	Tele:31-20-565-3209 Fax:31-20-972318	Amsterdam
Norway	Paul Oyan	Tele:47-2-99-93-83 Fax:47-2-99-93-33	Oslo
Pakistan	Inayat Ali Ebrahim	Tele:92-21-525181-190	Karachi
Portugal	Maria Augusta Morgado	Tele:351-1-542377 x3228	Lisbon
Saudi Arabia	Firas Halawani	Tele:966-2-6600007	Jeddah
South Africa	Anita Volker	Tele:27-11-224-9111	Joh'burg
Spain	Jose Maria Castro	Tele:34-1-397-9502 Fax:34-1-556-8352	Madrid
Sweden	Anna Rodholm	Tele:46-8-793-1000 x1403 Fax:46-8-793-1020	Stockholm

Figure 1. List of OS/2 Countries and Contacts (Continued)



Country	Contact	Details	Location
Europe, Middle East and Africa Corporation (EMEA) (Continued)			
Switzerland	Joerg Henseleit	Tele:41-1-207-2725 Fax:41-1-201-2228	Zurich
Turkey	Aysin Berkman	Tele:90-1-1800900 Fax:90-1-1780437	Istanbul
United Kingdom	Gordon Bell	Tele:44-256-475050 x8396	Basingstoke
Yugoslavia	Drago Kodele	Tele:38-61-325461	Ljubljana
ROECE*	Carolyn Sandner	Tele:43-222-21145 x6842 Fax:43-222-21145-3102	Vienna

*Eastern European countries not including the USSR.

Figure 1. List of OS/2 Countries and Contacts

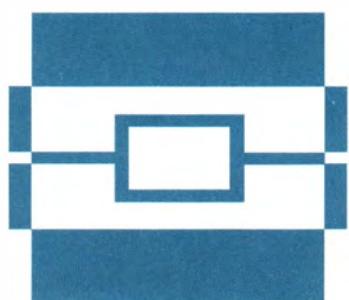
We welcome any comments and suggestions for international topics or article submissions for the *Developer* magazine. Please contact me with your ideas.

George Reader, IBM Europe, 6th Floor Mountbatten House, Basing View, Basingstoke, Hampshire RG21 1EJ, England. Telephone: 44-256-475050 Extension 8052 Fax: 44-256-58684. Mr. Reader is a product business manager working in a system software group having product management and support and marketing support responsibilities for EMEA in the Personal Systems Business Unit. He joined IBM as a customer engineer in 1968. Before leaving CE to join the PC/PS business in 1985, he held various positions in the field and in staff/support functions.



Systems Application Architecture

Multi-Phased Cooperative Processing Application Development



by Kevin Kornfeld

In a perfect world, application designers working towards cooperative processing solutions would design optimal systems from the ground up, re-architecting where necessary to achieve the best level of performance, user friendliness, and resource exploitation possible in the existing technological environment. All this, while positioning to make full use of any and all future developments expected on the distant horizon. And of course, such accomplishments would be achieved at an absolute minimum cost in terms of time, staff, market presence, and money.

Occasionally, a development team may have this sort of luxury. Unfortunately, there are far too many real-world constraints that prevent easy decisions on complex issues spanning the worlds of marketing and technology.

A typical scenario is that of a vendor or customer shop with a host-based (MVS, VM or AS/400®) application. The decision must be made as to whether or not this application should evolve into a cooperative processing implementation. Justifications generally range from a desire for a graphical user interface (GUI) based on SAA's Common User Access (CUA) architecture, to a dynamic marketing opportunity due to the increasingly omnipresent nature of personal computer workstations in the office. But no one is usually willing to scrap existing 3270 or 5250 terminal inventories.

To address this concern, a multi-phased development strategy is proposed to alleviate some of the most pressing concerns, while still focusing on a sound, long-term strategy best suited to address the universal need for both timeliness and quality.

COOPERATIVE PROCESSING ENHANCEMENTS

Cooperative processing refers to the concept of two processors, one of which is workstation based, working together in a coordinated manner to share resources or functionality. The term *added-value* is used to indicate improvements to the original host application derived from its evolution to cooperative processing.

Some of the most common *added-value* improvements are:

- The addition of a windowed, graphical user interface environment
- Conformance to SAA Common User Access (CUA)



Kevin Kornfeld



	Enhancement	Detailed Description
1	windowed, graphical user interface	use of OS/2 Presentation Mgr "WIN" & "GPI" interface
2	CUA conformance	SAA Common User Access: Graphical or Workplace models
3	business graphics	use of pie charts, bar graphs, etc. as supplement to tabular data on host application
4	bitmap images	bits representing graphical images, PM "GpiLoadBitmap" – allow iconic user interaction
5	local data/file storage	use of OS/2 Database Manager or local file storage to keep supplemental data records not normally maintained by host
6	enhanced field validation	verification of input entry fields for numeric, date/time specifications, or customized formats (prevents sending of incorrect data to host)
7	help and tutorial	use of OS/2 PM Information Presentation Facility (IPF) – can supplement or update existing host on-line help, and/or add workstation-based user tutorial feature
8	multi-processing	through this and multi-threading, make full use of the multi-tasking capabilities of OS/2 to increase user productivity and application performance capacity
9	distributed data	workstation-based: use OS/2 Database Manager Remote Data Services (RDS) to gain LAN data location transparency host-workstation: use APPC for SQL command transport and/or data duplication or transportation
10	distributed functionality	workstation-based: use OS/2 Named Pipes and/or APPC to share application services or transactions host-workstation: use APPC to share application services or transactions

Figure 1. Cooperative Processing Added-Value

- Use of business graphics
- Use of locally (or LAN) accessed images (bitmaps)
- Local data/file storage
- Enhanced field validation
- Enhanced Help and Tutorial facilities
- Multi-processing for improved performance, user responsiveness, and productivity
- Distributed data (workstation-based or host-workstation)
- Distributed functionality (workstation-based or host-workstation)
- APPC communications skills
- "C" language programming skills
- LAN skills, new programming methodologies for OS/2
- OS/2 Presentation Manager skills
- OS/2 Communications Manager configuration skills
- OS/2 Database Manager (and Remote Database Services) configuration skills
- Re-architecture of the host application
- Familiarity with SAA Common User Access architectural conventions
- New equipment and/or staffing requirements

Figure 1 is a table summarizing the above, and includes more detail as to the nature of each added-value.

The success of the business case for pursuing a cooperative processing strategy is proportional to the incorporation of these benefits into the project proposal. Yet rigorous implementation of all enhancements generally requires that a number of activities occur, often simultaneously. These may combine to the detriment of the business case, especially in terms of time and cost. Some of these include the development of:



The success of the business case for pursuing a cooperative processing strategy is proportional to the incorporation of these benefits into the project proposal

PHASE I

The objective of a *Phase I* project is to obtain the cooperative processing benefits, while incurring as few costs as possible. One major cost is in acquiring the required skill set for cooperative processing (as outlined above). All of these skills should eventually be learned; but this can be accomplished gradually. At least part of the learning curve and/or implementation time for the later

phases of development can be subsidized by the Phase I product revenue stream. The first phase is used to establish a market share and income, while later phases are used to provide product or performance enhancements.

The most time and cost efficient alternative for Phase I development is the use of a *front-end* to the host application. There are a number of available software tools or

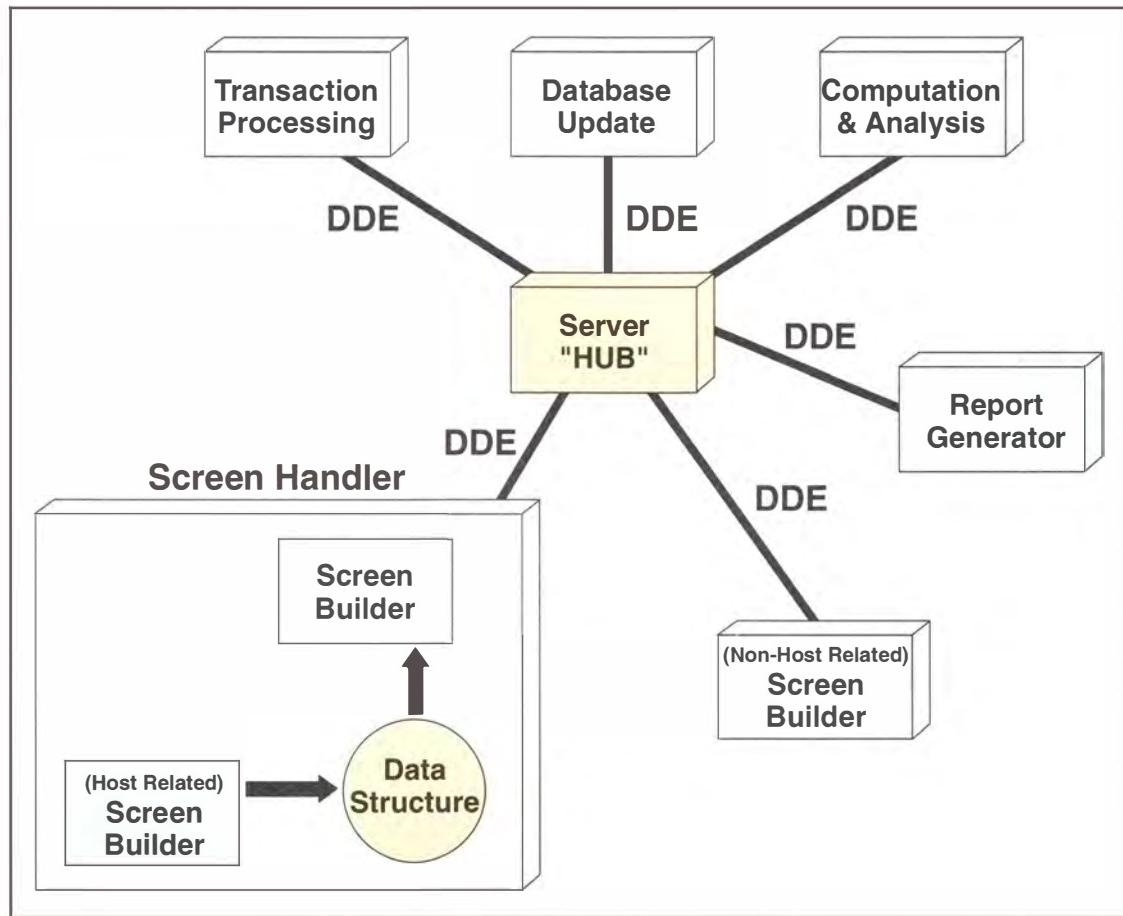


Figure 2. Server Process "HUB"



Server Process	Peripheral Process 1 (provides service "AAA")	Peripheral Process 2 (provides service "BBB")	Peripheral Process n (provides service "ZZZ")
Register as DDE server			
"Launch" each process	Default active process		
Verify each process exists (exit on error)	● ● ●	User requests service "BBB" (notify server)	
Activate Process 2		Process activated ● ● ●	
Activate Process n		User requests service "ZZZ" (notify server)	Process activated ● ●

Figure 3. Peripheral Process Activation



enablers which can simplify this development process; alternatively, the front-end can be implemented directly via "C" or COBOL using OS/2 Communications Manager EHLLAPI (Emulator High Level Language Application Programming Interface). Regardless of the tool or methodology, the concept of front-ending is the same: to retain the host application *as-is*, while providing a new graphical user *face* with the workstation. The host screens are, in effect, intercepted at the workstation, then presented in an entirely revised format to the user. Keystrokes required by the host software are passed to the host screen by the workstation application directly, not by the user, who is instead interfacing to the screen presentation of the workstation. The net effect is that the host application remains unchanged, and is unaware that anything is different about this user. The individual at the workstation, on the other hand, is interfacing to a set of screens which have no DIRECT correlation to the original application screens. The intent is for the new workstation screens to be more intuitive, easier to use, and in conformance with the SAA user interface architecture of CUA.

Use of EASEL WorkbenchTM, an enabler tool by Easel Corporation of Burlington, MA, for this phase of development is recommended for the following reasons:

- The learning curve is extremely short. Therefore, the time it takes to develop and code an application is greatly reduced (versus native "C" or COBOL coding).

- The EASEL/2TM component of EASEL Workbench gives the developer a way to utilize the OS/2 Presentation Manager, the OS/2 Database Manager, and OS/2 Communications Manager EHLLAPI and LU6.2 (APPN) protocols.
- EASEL/2 supports Dynamic Data Exchange (DDE), thus enabling inter-process communications for the OS/2 Presentation Manager.
- EASEL/2 additionally provides a large number of development tools (such as a Business Graphics package) for the programmer's use.

With EASEL/2, the first eight *added-value* benefits of cooperative processing (referring back to Figure 1) can be realized in a relatively short period of time. The final two may be partially accomplished via a *workstation-based* environment.

EASEL/2 usage also minimizes the associated costs, since the speed and level of skill acquisition in other areas, such as "C" programming and the OS/2 Presentation Manager, can be delayed over a longer period of time.

Whether using the EASEL language or any other developmental tool, it is generally considered prudent to architect the application in a manner consistent with the goal of eventually transforming it from a simple *front-end* to that of a more robust cooperative processing software solution. That is, a solution implementing the final two *added-value* benefits of distributed data and distributed functionality in the host-workstation environments.

One such recommended architectural technique is to employ an invisible server process to act as a *hub* for a set of peripheral processes, each with individual functions or tasks (see Figure 2). These processes would control such operations as computation and analysis, report generation, database update, transaction processing, screen presentation, and other operations, depending upon the application. Use of a server hub keeps direction of the multi-processing environment centralized, making change relatively easy to control, and enforcing the mutual independence of each peripheral process. This makes alteration of these peripheral processes relatively trouble-free, since any change would at most affect only the server hub process where the interface is handled.

The host-related screen handler would be divided into separate tasks for *read* and *build*. A screen *reader*, using EHLLAPI to capture the host screen in an OS/2 Presentation Manager presentation space, would be responsible for setting up a data structure for input to the screen *builder*. This data structure would then be used for construction of the actual user CUA screen(s). One data structure might be used for multiple screens, or multiple structures might be required for just one screen — this would be application dependent. The long-term strategy would be for the *builder* to rely only on the data structure, making the use of EHLLAPI to set up the structure within the screen *reader* irrelevant. This becomes important in Phase II of the project.

Note that an entire system can be constructed using EASEL/2. The various processes would be controlled by the server hub via DDE. Control could shift from one process to the next under the control of the server process (see Figure 3 for a high-level illustration of this protocol). Two or more processes could be designed to perform in parallel. There would be no requirement that all processes be written in the EASEL language; indeed, the only requirement would be that all were OS/2 Presentation Manager applications with DDE support provided.



The concept of multi-phase application development assumes that while one set of developers is implementing one phase, another set is preparing for the next



PHASE II

In order to introduce the final two added-value benefits of distributed data and function for the host-workstation mode, it becomes necessary to use Advanced Program to Program Communications (APPC). APPC is an implementation of IBM's LU6.2 SNA architecture for peer to peer communications. APPC essentially allows workstations to either initiate or accept communications via *conversations* from other network residents either other workstations or host machines. All are considered as equals, or *peers*, on the network.

The concept of multi-phase application development assumes that while one set of developers is implementing one phase, another set is preparing for the next. Thus, when Phase I is ready for release, APPC skills should already exist in-house. The question therefore becomes one of how best to accomplish the distribution of data and function via APPC, assuming this skill-base to now be available.

Ideally, data distribution on an SAA network would be completely transparent to the programmer. Thus, an SQL command, such as RETRIEVE, would not care if the database table it referenced were located locally (on the workstation) or somewhere else on the LAN. This would be a simple matter of configuration details, left to an administrator. For LAN based applications (that is, no host-based database interaction), this transparency is currently available via the OS/2 Database Manager Remote Database Services (RDS). However, to share host data, a transport mechanism would have to be written by the application developer. This transport would probably be accomplished via APPC, since it is supported on all SAA host platforms.

APPC could also be used to distribute application services and transactions. It would also be employed in place of direct host screen manipulation. That is, the host software transfers whatever information it wishes to the workstation via APPC; the workstation displays the appropriate CUA screen; information is then retrieved, and returned to the host. Referring back to Figure 2, the EHLLAPI oriented *reader* program can now be replaced with a corresponding APPC *reader*. The same data structure would be set up by both, which would then be used as input to the screen *builder*, as before. The amount of code changed is minimal. There is no affect on the screens the user sees, unless this is desired.

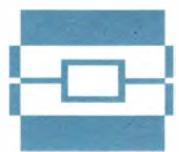
With this usage of APPC for host-based interaction, it must be remembered that a change to the host application will now be required. This change may or may not require a complete re-architecture, depending upon the original design. Ideally, the host software will separate display input/output from functional or transaction routing. If so, the display or screen module can be paralleled by a corresponding APPC module. For non-workstation users, the host screen handler remains in control. Workstation users are handled via the APPC module instead. Regardless, the routing mechanism and function dispatch code remains identical.

This host APPC module could also handle data distribution and associated processing, as required.

PHASE III

At this stage, the application program may still be *tool* based, in that it may not be written entirely in "C" Presentation Manager. However, it should still provide all ten of the *added-value* cooperative processing enhancements from Figure 1. There may of course be additional *added-value* features as well, due to application or industry specific requirements or conditions. This is now a critical juncture for the application product. The following strategic questions must now be addressed:

- Are there new technologies that need to be incorporated into the existing product?
- Would it be possible to integrate this existing application with an other host and/or workstation-based application? This should be technologically feasible via the multiple-session support included with the OS/2 Communications Manager where multiple sessions on the same host, and/or single or multiple sessions on multiple hosts, may be simultaneously accessed and via OS/2 multi-processing.
- Is the performance of the code adequate for projected future usage requirements? That is, will it be necessary to convert from an enabler-base to a "C" Presentation Manager application? Of course, this decision may be made on a process-by-process basis. Changing one peripheral process does not affect the others, so long as the Presentation Manager DDE interface is retained.
- Are there additional capabilities not yet implemented? Is there a business case for such an implementation?





ACKNOWLEDGEMENTS

The author wishes to thank the following people for their assistance in the preparation of this article: Eleonora Shafir, Steven Poole, Kevin Gilhooly, Bill Boland, and Harlon Trowbridge.

Kevin Kornfeld, IBM, US Marketing and Services, 9 Village Circle, Roanoke, TX 76262. Mr. Kornfeld joined IBM in April of 1989 as an Advisory Marketing Support representative. A member of the Software Vendor Systems Center, he works with select software vendors in developing SAA applications for the OS/2 platform, providing both design consulting and vendor education. He developed (and is lead instructor for) the SAA Express program offering "Cooperative Processing on OS/2", a five-day class discussing cooperative processing concepts and the use of EASEL as a programming development tool. Mr. Kornfeld has a BSEE degree from the University of Arizona in Tucson, and a Master of Computer Science from Southern Methodist University in Dallas, Texas.

IBM
®



© IBM Corporation
All Rights Reserved

International Business Machines Corporation
P.O. Box 1328
Boca Raton, FL 33429-1328

G362-0001-10

